CSHL 2010 Course Materials

Release 1.0

Sean Davis

June 28, 2010

CONTENTS

Contents:

CHAPTER

ONE

CONNECTING TO PUBLIC DATA RESOURCES

Author Sean Davis

Date 2010-06-28

Contact sdavis2@mail.nih.gov

1.1 Introduction

The biological sciences have quickly transitioned toward being a more and more data-intensive science. High-throughput assays such as microarrays, second-generation sequencing platforms, and proteomic assays now play key roles at nearly every level of research from the most basic science through to the most clinical research. In fact, high-throughput technologies are showing up in clinical practice, directly affecting patient care. Many of the data that have been generated to drive hypothesis generation and hypothesis-driven research have been deposited in public repositories. For microarray data, the largest two repositories are the NCBI Gene Expression Omnibus (GEO) and the EBI ArrayExpress database. For sequencing data, the largest public repository is the Sequence Read Archive (SRA), a joint project among the NCBI, the EBI, and DDBJ. Accessing these treasure troves is part of the scientific process for many experiments. In fact, in some cases, an experiment has already been done that either directly or indirectly answers a question of interest to another investigator; simply downloading and reanalyzing the results may be the most efficient way forward. In other cases, an understanding of systematic biases or other operating characteristics of an assay is best accomplished when vast quantities of data generated by the assay are examined simultaneously. Whatever the reason, facile queries of and access to these data repositories is becoming increasingly important. Several bioconductor packages allow querying and access to data sets of interest from within R, making further processing easier.

This vignette provides some examples of using tools to access the NCBI GEO data and metadata using the GEOquery and GEOmetadb packages. Also available are the ArrayExpress package package for accessing ArrayExpress data and the SRAdb package package for both querying SRA metadata and getting access to raw data in a convenient way. For more details on SRAdb and ArrayExpress, see the package documentation; both include extensive examples.

1.2 NCBI GEO

1.2.1 Overview of GEO

The NCBI Gene Expression Omnibus (GEO) serves as a public repository for a wide range of high-throughput experimental data. These data include single and dual channel microarray-based experiments measuring mRNA, genomic

DNA, and protein abundance, as well as non-array techniques such as serial analysis of gene expression (SAGE), mass spectrometry proteomic data, and high-throughput sequencing data.

At the most basic level of organization of GEO, there are four basic entity types. The first three (Sample, Platform, and Series) are supplied by users; the fourth, the dataset, is compiled and curated by GEO staff from the user-submitted data [1].

Platforms

A Platform record describes the list of elements on the array (e.g., cDNAs, oligonucleotide probesets, ORFs, antibodies) or the list of elements that may be detected and quantified in that experiment (e.g., SAGE tags, peptides). Each Platform record is assigned a unique and stable GEO accession number (GPLxxx). A Platform may reference many Samples that have been submitted by multiple submitters.

Samples

A Sample record describes the conditions under which an individual Sample was handled, the manipulations it underwent, and the abundance measurement of each element derived from it. Each Sample record is assigned a unique and stable GEO accession number (GSMxxx). A Sample entity must reference only one Platform and may be included in multiple Series.

Series

A Series record defines a set of related Samples considered to be part of a group, how the Samples are related, and if and how they are ordered. A Series provides a focal point and description of the experiment as a whole. Series records may also contain tables describing extracted data, summary conclusions, or analyses. Each Series record is assigned a unique and stable GEO accession number (GSExxx). Series records are available in a couple of formats which are handled by GEOquery independently. The smaller and newer GSEMatrix files are quite fast to parse; a simple flag is used by GEOquery to choose to use GSEMatrix files (see below).

Datasets

GEO DataSets (GDSxxx) are curated sets of GEO Sample data. A GDS record represents a collection of biologically and statistically comparable GEO Samples and forms the basis of GEO's suite of data display and analysis tools. Samples within a GDS refer to the same Platform, that is, they share a common set of probe elements. Value measurements for each Sample within a GDS are assumed to be calculated in an equivalent manner; that is, considerations such as background processing and normalization are consistent across the dataset. Information reflecting experimental design is provided through GDS subsets.

1.2.2 Getting Started using GEOquery package

Getting data from GEO into R is really quite easy. There is only one command that is needed, getGEO. This one function interprets its input to determine how to get the data from GEO and then parse the data into useful R data structures. Usage is quite simple:

```
> library(GEOquery)
```

This loads the GEOquery library.

[1] See http://www.ncbi.nih.gov/geo for more information.

```
> gds <- getGEO(filename = system.file("extdata/GDS507.soft.gz",
+     package = "GEOquery"))
```

Now, the R object gds contains the R data structure (of class GDS) that represents the GDS507 entry from GEO. You'll note that the filename used to store the download was output to the screen (but not saved anywhere) for later use to a call to getGEO(filename=...).

We can do the same with any other GEO accession, such as GSM3, a GEO sample.

```
> gsm <- getGEO(filename = system.file("extdata/GSM11805.txt.gz",
+     package = "GEOquery"))
```

GEOquery Data Structures

The GEOquery data structures really come in two forms. The first, comprising GDS, GPL, and GSM all behave similarly and accessors have similar effects on each. The fourth GEOquery data structure, GSE is a composite data type made up of lists of GSM and GPL objects and will not be described in more detail; refer to the GEOquery documentation for more details.

Each of the GPL, GSM, and GDS classes is comprised of a metadata header (taken nearly verbatim from the SOFT format header) and a GEODataTable. The GEODataTable has two simple parts, a Columns part which describes the column headers on the Table part. There is also a show method for each class. For example, using the gsm from above:

```
> Meta(gsm)
$channel_count
[1] "1"

$comment
[1] "Raw data provided as supplementary file"

$contact_address
[1] "715 Albany Street, E613B"

$contact_city
[1] "Boston"

$contact_country
[1] "USA"

$contact_department
[1] "Genetics and Genomics"

$contact_email
[1] "mlenburg@bu.edu"

$contact_fax
[1] "617-414-1646"

$contact_institute
[1] "Boston University School of Medicine"

$contact_name
[1] "Marc,E.,Lenburg"

$contact_phone
```

```
[1] "617-414-1375"

$contact_state
[1] "MA"

$contact_web_link
[1] "http://gg.bu.edu"

$`contact_zip/postal_code`
[1] "02130"

$data_row_count
[1] "22283"

$description
[1] "Age = 70; Gender = Female; Right Kidney; Adjacent Tumor Type = clear cell; Adjacent Tumor Fuhrma
[2] "Keywords = kidney"
[3] "Keywords = renal"
[4] "Keywords = RCC"
[5] "Keywords = carcinoma"
[6] "Keywords = cancer"
[7] "Lot batch = 2004638"

$geo_accession
[1] "GSM11805"

$last_update_date
[1] "May 28 2005"

$molecule_ch1
[1] "total RNA"

$organism_ch1
[1] "Homo sapiens"

$platform_id
[1] "GPL96"

$series_id
[1] "GSE781"

$source_name_ch1
[1] "Trizol isolation of total RNA from normal tissue adjacent to Renal Cell Carcinoma"

$status
[1] "Public on Nov 25 2003"

$submission_date
[1] "Oct 20 2003"

$supplementary_file
[1] "ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSM11nnn/GSM11805/GSM11805.CEL.gz"

$title
[1] "N035 Normal Human Kidney U133A"

$type
[1] "RNA"
```

```
> Table(gsm)[1:5, ]
          ID_REF  VALUE ABS_CALL
1 AFFX-BioB-5_at  953.9        P
2 AFFX-BioB-M_at 2982.8        P
3 AFFX-BioB-3_at 1657.9        P
4 AFFX-BioC-5_at 2652.7        P
5 AFFX-BioC-3_at 2019.5        P
> Columns(gsm)
    Column
1   ID_REF
2    VALUE
3 ABS_CALL
                                                              Description
1
2                         MAS 5.0 Statistical Algorithm (mean scaled to 500)
3 MAS 5.0 Absent, Marginal, Present call  with Alpha1 = 0.05, Alpha2 = 0.065
```

The GPL class behaves exactly as the GSM class (i.e., it has the same methods). However, the GDS has a bit more information associated with the Columns method:

```
> Columns(gds)
     sample disease.state individual
1  GSM11815           RCC        035
2  GSM11832           RCC        023
3  GSM12069           RCC        001
4  GSM12083           RCC        005
5  GSM12101           RCC        011
6  GSM12106           RCC        032
7  GSM12274           RCC          2
8  GSM12299           RCC          3
9  GSM12412           RCC          4
10 GSM11810        normal        035
11 GSM11827        normal        023
12 GSM12078        normal        001
13 GSM12099        normal        005
14 GSM12269        normal          1
15 GSM12287        normal          2
16 GSM12301        normal          3
17 GSM12448        normal          4

1             Value for GSM11815: C035 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
2             Value for GSM11832: C023 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
3             Value for GSM12069: C001 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
4             Value for GSM12083: C005 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
5             Value for GSM12101: C011 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
6             Value for GSM12106: C032 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
7               Value for GSM12274: C2 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
8               Value for GSM12299: C3 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
9               Value for GSM12412: C4 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of tot
10      Value for GSM11810: N035 Normal Human Kidney U133B; src: Trizol isolation of total RNA from n
11      Value for GSM11827: N023 Normal Human Kidney U133B; src: Trizol isolation of total RNA from n
12      Value for GSM12078: N001 Normal Human Kidney U133B; src: Trizol isolation of total RNA from n
13      Value for GSM12099: N005 Normal Human Kidney U133B; src: Trizol isolation of total RNA from n
14        Value for GSM12269: N1 Normal Human Kidney U133B; src: Trizol isolation of total RNA from n
15 Value for GSM12287: N2 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA from n
16 Value for GSM12301: N3 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA from n
17 Value for GSM12448: N4 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA from n
```

Converting to BioConductor ExpressionSets and limma MALists

GEO datasets are (unlike some of the other GEO entities), quite similar to the limma data structure MAList and to the Biobase data structure ExpressionSet. Therefore, there are two functions, GDS2MA and GDS2eSet that accomplish these conversions.

Converting GDS to an ExpressionSet

Taking our gds object from above, we can simply do:

```
> eset <- GDS2eSet(gds, do.log2 = TRUE)
File stored at:
/var/folders/F+/F+PwkbXqF6WeunvinD8pZk+++TI/-Tmp-//RtmpGNZvHN/GPL97.annot
```

Now, eset is an ExpressionSet that contains the same information as in the GEO dataset, including the sample information, which we can see here:

```
> eset
ExpressionSet (storageMode: lockedEnvironment)
assayData: 22645 features, 17 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM11815, GSM11832, ..., GSM12448  (17 total)
  varLabels and varMetadata description:
    sample: NA
    disease.state: NA
    individual: NA
    description: NA
featureData
  featureNames: 200000_s_at, 200001_at, ..., AFFX-TrpnX-M_at  (22645 total)
  fvarLabels and fvarMetadata description:
    ID: ID from Platform data table
    Gene.title: Entrez Gene name
    ...: ...
    GO.Component.1: Gene Ontology Component identifier
    (21 total)
  additional fvarMetadata: Column
experimentData: use 'experimentData(object)'
  pubMedIds: 14641932
Annotation:
> pData(eset)
           sample disease.state individual
GSM11815 GSM11815           RCC        035
GSM11832 GSM11832           RCC        023
GSM12069 GSM12069           RCC        001
GSM12083 GSM12083           RCC        005
GSM12101 GSM12101           RCC        011
GSM12106 GSM12106           RCC        032
GSM12274 GSM12274           RCC          2
GSM12299 GSM12299           RCC          3
GSM12412 GSM12412           RCC          4
GSM11810 GSM11810        normal        035
GSM11827 GSM11827        normal        023
GSM12078 GSM12078        normal        001
GSM12099 GSM12099        normal        005
```

```
GSM12269 GSM12269        normal          1
GSM12287 GSM12287        normal          2
GSM12301 GSM12301        normal          3
GSM12448 GSM12448        normal          4

GSM11815            Value for GSM11815: C035 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM11832            Value for GSM11832: C023 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12069            Value for GSM12069: C001 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12083            Value for GSM12083: C005 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12101            Value for GSM12101: C011 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12106            Value for GSM12106: C032 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12274              Value for GSM12274: C2 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12299              Value for GSM12299: C3 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM12412              Value for GSM12412: C4 Renal Clear Cell Carcinoma U133B; src: Trizol isolation
GSM11810      Value for GSM11810: N035 Normal Human Kidney U133B; src: Trizol isolation of total RNA
GSM11827      Value for GSM11827: N023 Normal Human Kidney U133B; src: Trizol isolation of total RNA
GSM12078      Value for GSM12078: N001 Normal Human Kidney U133B; src: Trizol isolation of total RNA
GSM12099      Value for GSM12099: N005 Normal Human Kidney U133B; src: Trizol isolation of total RNA
GSM12269        Value for GSM12269: N1 Normal Human Kidney U133B; src: Trizol isolation of total RNA
GSM12287 Value for GSM12287: N2 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA
GSM12301 Value for GSM12301: N3 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA
GSM12448 Value for GSM12448: N4 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA
```

Converting GDS to an MAList

No annotation information (called platform information by GEO) was retrieved from because ExpressionSet does not contain slots for gene information, typically. However, it is easy to obtain this information. First, we need to know what platform this GDS used. Then, another call to getGEO will get us what we need.

```
> Meta(gds)$platform
[1] "GPL97"
> gpl <- getGEO(filename = system.file("extdata/GPL97.annot.gz",
+     package = "GEOquery"))
```

So, gpl now contains the information for GPL5 from GEO. Unlike the ExpressionSet class, the limma MAList class does store gene annotation information, so we can use our newly created gpl object of class GPL in a call to GDS2MA like so:

```
> MA <- GDS2MA(gds, GPL = gpl)
> MA
An object of class "MAList"
$M
     GSM11815 GSM11832 GSM12069 GSM12083 GSM12101 GSM12106 GSM12274 GSM12299
[1,]   4254.0   5298.2   4026.5   3498.4   3566.4   4903.1   6372.6   4829.1
[2,]  17996.2  12010.7  10283.5   2534.7  11048.4  13354.0   8563.8  17247.6
[3,]  41678.8  39116.9  38758.9  32847.7  39633.9  43511.2  46856.7  47032.4
[4,]  65390.9  34806.2  31257.2  28308.5  67447.5  56989.9  57972.5  57570.5
[5,]  19030.1  15813.6  16355.7   9579.7  14273.5  17217.0  19116.9  17487.6
     GSM12412 GSM11810 GSM11827 GSM12078 GSM12099 GSM12269 GSM12287 GSM12301
[1,]   5205.8   2756.8   3932.0   3729.9   3223.4   3640.5   4886.3   4070.2
[2,]  16018.5   6077.0  15703.8  10138.5  11614.4   8460.5  10282.6  11844.3
[3,]  22152.2  26660.7  26373.6  23809.6  24749.3  21936.8  31462.8  22733.7
[4,]  29062.2  35140.9  23629.3  22100.5  21651.0  18550.7  23496.5  21315.4
[5,]  14671.6  17733.1  18022.4  17957.4  15958.0  15799.8  16685.8  18817.3
     GSM12448
```

```
[1,]   3482.1
[2,]   9741.6
[3,]  25395.5
[4,]  28631.4
[5,]  17421.1
22640 more rows ...

$A
NULL

$targets
    sample disease.state individual
1 GSM11815           RCC        035
2 GSM11832           RCC        023
3 GSM12069           RCC        001
4 GSM12083           RCC        005
5 GSM12101           RCC        011

1 Value for GSM11815: C035 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA from
2 Value for GSM11832: C023 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA from
3 Value for GSM12069: C001 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA from
4 Value for GSM12083: C005 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA from
5 Value for GSM12101: C011 Renal Clear Cell Carcinoma U133B; src: Trizol isolation of total RNA from
12 more rows ...

$genes
           ID                                                 Gene.title
1 200000_s_at PRP8 pre-mRNA processing factor 8 homolog (S. cerevisiae)
2   200001_at                                  calpain, small subunit 1
3   200002_at                                     ribosomal protein L35
4 200003_s_at                                     ribosomal protein L28
5   200004_at       eukaryotic translation initiation factor 4 gamma, 2
  Gene.symbol Gene.ID UniGene.title UniGene.symbol UniGene.ID
1       PRPF8   10594
2      CAPNS1     826
3       RPL35   11224
4       RPL28    6158
5      EIF4G2    1982
                                                                                                   Nucleotide.Tit
1                   Homo sapiens PRP8 pre-mRNA processing factor 8 homolog (S. cerevisiae) (PRPF8), mR
2                             Homo sapiens calpain, small subunit 1 (CAPNS1), transcript variant 1, mR
3                                                    Homo sapiens ribosomal protein L35 (RPL35), mR
4                                                    Homo sapiens ribosomal protein L28 (RPL28), mR
5 Homo sapiens eukaryotic translation initiation factor 4 gamma, 2 (EIF4G2), transcript variant 1, mR
         GI GenBank.Accession Platform_CLONEID Platform_ORF Platform_SPOTID
1  91208425         NM_006445             <NA>         <NA>            <NA>
2  51599152         NM_001749             <NA>         <NA>            <NA>
3  78190471         NM_007209             <NA>         <NA>            <NA>
4  34486095         NM_000991             <NA>         <NA>            <NA>
5 111494227         NM_001418             <NA>         <NA>            <NA>
  Chromosome.location
1            17p13.3
2           19q13.12
3             9q34.1
4            19q13.4
5              11p15
                                        Chromosome.annotation
1     Chromosome 17, NC_000017.9 (1500673..1534926, complement)
```

```
2              Chromosome 19, NC_000019.8 (41322758..41333095)
3 Chromosome 9, NC_000009.10 (126659979..126664061, complement)
4              Chromosome 19, NC_000019.8 (60589112..60595265)
5  Chromosome 11, NC_000011.8 (10775169..10787158, complement)

1                         RNA binding///RNA splicing factor activity, transesterification mechanism//
2                      calcium ion binding///calcium-dependent cysteine-type endopeptidase activity//
3                                                 mRNA binding///protein binding///structural consti
4            RNA binding///protein binding///structural constituent of ribosome///structural consti
5 protein binding///protein binding///translation initiation factor activity///translation initiatio

1 RNA splicing///nuclear mRNA splicing, via spliceosome///nuclear mRNA splicing, via spliceosome///re
2                                                                                                po
3
4
5                                           RNA metabolic process///cell cycle arrest///cell death//
                                                                 GO.Component
1                                  nuclear speck///nucleus///snRNP U5///spliceosome
2                                                     cytoplasm///plasma membrane
3 cytosol///cytosolic large ribosomal subunit///intracellular///nucleolus///ribosome
4            cytosol///cytosolic large ribosomal subunit///intracellular///ribosome
5                               eukaryotic translation initiation factor 4F complex
                                     GO.Function.1
1             GO:0003723///GO:0031202///GO:0005515
2             GO:0005509///GO:0004198///GO:0005515
3             GO:0003729///GO:0005515///GO:0003735
4 GO:0003723///GO:0005515///GO:0003735///GO:0003735
5 GO:0005515///GO:0005515///GO:0003743///GO:0003743
                                                   GO.Process.1
1 GO:0008380///GO:0000398///GO:0000398///GO:0050896///GO:0007601
2                                                    GO:0008284
3                                                    GO:0006414
4                                        GO:0006412///GO:0006414
5             GO:0016070///GO:0007050///GO:0008219///GO:0006446
                                                 GO.Component.1
1             GO:0016607///GO:0005634///GO:0005682///GO:0005681
2                                        GO:0005737///GO:0005886
3 GO:0005829///GO:0022625///GO:0005622///GO:0005730///GO:0005840
4             GO:0005829///GO:0022625///GO:0005622///GO:0005840
5                                                    GO:0016281
22640 more rows ...

$notes
$channel_count
[1] "1"

$dataset_id
 [1] "GDS507" "GDS507" "GDS507" "GDS507" "GDS507" "GDS507" "GDS507" "GDS507"
 [9] "GDS507" "GDS507" "GDS507" "GDS507"

$description
 [1] "Investigation into mechanisms of renal clear cell carcinogenesis (RCC). Comparison of renal cl
 [2] "RCC"
 [3] "normal"
 [4] "035"
 [5] "023"
 [6] "001"
 [7] "005"
 [8] "011"
 [9] "032"
[10] "1"
[11] "2"
[12] "3"
[13] "4"

$email
[1] "geo@ncbi.nlm.nih.gov"

$feature_count
[1] "22645"

$institute
[1] "NCBI NLM NIH"

$name
[1] "Gene Expression Omnibus (GEO)"

$order
[1] "none"

$platform
[1] "GPL97"

$platform_organism
[1] "Homo sapiens"

$platform_technology_type
[1] "in situ oligonucleotide"

$pubmed_id
[1] "14641932"

$ref
[1] "Nucleic Acids Res. 2005 Jan 1;33 Database Issue:D562-6"

$reference_series
[1] "GSE781"

$sample_count
[1] "17"

$sample_id
 [1] "GSM11815,GSM11832,GSM12069,GSM12083,GSM12101,GSM12106,GSM12274,GSM12299,GSM12412"
 [2] "GSM11810,GSM11827,GSM12078,GSM12099,GSM12269,GSM12287,GSM12301,GSM12448"
 [3] "GSM11810,GSM11815"
 [4] "GSM11827,GSM11832"
 [5] "GSM12069,GSM12078"
 [6] "GSM12083,GSM12099"
 [7] "GSM12101"
 [8] "GSM12106"
 [9] "GSM12269"
[10] "GSM12274,GSM12287"
[11] "GSM12299,GSM12301"
[12] "GSM12412,GSM12448"

$sample_organism
```

```
[1] "Homo sapiens"

$sample_type
[1] "RNA"

$title
[1] "Renal clear cell carcinoma (HG-U133B)"

$type
 [1] "gene expression array-based" "disease state"
 [3] "disease state"               "individual"
 [5] "individual"                  "individual"
 [7] "individual"                  "individual"
 [9] "individual"                  "individual"
[11] "individual"                  "individual"
[13] "individual"

$update_date
[1] "Mar 04 2004"

$value_type
[1] "count"

$web_link
[1] "http://www.ncbi.nlm.nih.gov/projects/geo"
```

Now, the R object MA is of class MAList and contains not only the data, but the sample information and gene information associated with GDS507.

Getting GSE Series Matrix files as an ExpressionSet

GEO Series are collections of related experiments. In addition to being available as SOFT format files, which are quite large, NCBI GEO has prepared a simpler format file based on tab-delimited text. The getGEO function can handle this format and will parse very large GSEs quite quickly. The data structure returned from this parsing is a list of ExpressionSets. As an example, we download and parse GSE2553.

```
> gse2553 <- getGEO("GSE2553", GSEMatrix = TRUE)
Found 1 file(s)
GSE2553_series_matrix.txt.gz
File stored at:
/var/folders/F+/F+PwkbXqF6WeunvinD8pZk+++TI/-Tmp-//RtmpGNZvHN/GPL1977.soft
> show(gse2553)
$GSE2553_series_matrix.txt.gz
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12600 features, 181 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM48681, GSM48682, ..., GSM48861  (181 total)
  varLabels and varMetadata description:
    title: NA
    geo_accession: NA
    ...: ...
    data_row_count: NA
    (27 total)
featureData
```

```
  featureNames: 1, 2, ..., 12600  (12600 total)
  fvarLabels and fvarMetadata description:
    ID: NA
    PenAt: NA
    ...: ...
    Chimeric_Cluster_IDs: NA
    (13 total)
  additional fvarMetadata: Column, Description
experimentData: use 'experimentData(object)'
Annotation: GPL1977
> show(pData(phenoData(gse2553[[1]]))[1:5, c(1, 6, 8)])
                                                                title type
GSM48681                      Patient sample ST18, Dermatofibrosarcoma  RNA
GSM48682                           Patient sample ST410, Ewing Sarcoma  RNA
GSM48683                            Patient sample ST130, Sarcoma, NOS  RNA
GSM48684 Patient sample ST293, Malignant Peripheral Nerve Sheath Tumor  RNA
GSM48685                             Patient sample ST367, Liposarcoma  RNA
                                source_name_ch1
GSM48681                   Dermatofibrosarcoma
GSM48682                         Ewing Sarcoma
GSM48683                          Sarcoma, NOS
GSM48684 Malignant Peripheral Nerve Sheath Tumor
GSM48685                           Liposarcoma
```

Accessing Raw Data from GEO

NCBI GEO accepts (but has not always required) raw data such as .CEL files, .CDF files, images, etc. Sometimes, it is useful to get quick access to such data. A single function, getGEOSuppFiles, can take as an argument a GEO accession and will download all the raw data associate with that accession. By default, the function will create a directory in the current working directory to store the raw data for the chosen GEO accession. Combining a simple sapply statement or other loop structure with getGEOSuppFiles makes for a very simple way to get gobs of raw data quickly and easily without needing to know the specifics of GEO raw data URLs.

Summary of GEOquery

The GEOquery package provides a bridge to the vast array resources contained in the NCBI GEO repositories. By maintaining the full richness of the GEO data rather than focusing on getting only the "numbers", it is possible to integrate GEO data into current Bioconductor data structures and to perform analyses on that data quite quickly and easily. These tools will hopefully open GEO data more fully to the array community at large.

1.2.3 Overview of the GEOmetadb package

One difficulty in dealing with GEO is finding the microarray data that is of interest. As part of the NCBI Entrez search system, GEO can be searched online via web pages or using NCBI Eutils. However, the web search is not as full-featured as it could be, particularly for programmatic access. NCBI Eutils offers another option for finding data within the vast stores of GEO, but it is cumbersome to use, often requiring multiple complicated Eutils calls to get at the relevant information. We have found it necessary to have ready access not just to the microarray data as provided by the GEOquery package, but also to the metadata describing the microarray experiments. The GEOmetadb package provides fast and programmatic access to GEO metadata.

The GEOmetadb package is an attempt to make querying the metadata describing microarray experiments, platforms, and datasets both easier and more powerful. At the heart of GEOmetadb is a SQLite database that stores nearly all the metadata associated with all GEO data types including GEO samples (GSM), GEO platforms (GPL), GEO data series (GSE), and curated GEO datasets (GDS), as well as the relationships between these data types. This database is generated on our server by parsing all the records in GEO and needs to be downloaded via a simple helper function to the user's local machine before GEOmetadb is useful. Once this is done, the entire GEO database is accessible with simple SQL-based queries. With the GEOmetadb database, queries that are simply not possible using NCBI tools or web pages are often quite simple. The relationships between the tables in the GEOmetadb SQLite database can be seen in figure below.

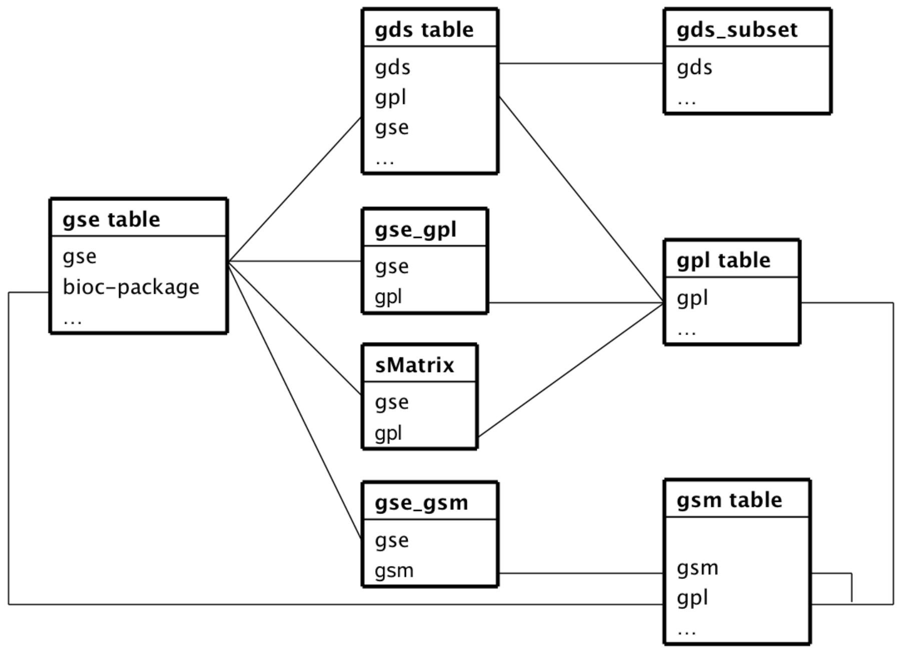


Figure 1.1: A graphical representation (sometimes called an *Entity-Relationship Diagram*) of the relationships between the tables in the GEOmetadb SQLite database.

We have faithfully parsed and maintained in GEO when creating GEOmetadb. This means that limitations inherent to GEO are also inherent in GEOmetadb. There has been no attempt to curate, semantically recode, or otherwise "clean up" GEO for GEOmetadb.

Also, GEOmetadb does not contain any microarray data. For access to microarray data from within R/Bioconductor. The expectation is that many users will find that the combination of GEOmetadb and GEOquery a natural one.

Conversion capabilities

A very typical problem for large-scale consumers of GEO data is to determine the relationships between various GEO accession types. As examples, consider the following questions:

- What samples are associated with GEO platform "GPL96", which represents the Affymetrix hgu133a array?

- What GEO Series were performed using "GPL96"?
- What samples are in my favorite three GEO Series records?
- How many samples are associated with the ten most popular GEO platforms?

Because these types of questions are common, GEOmetadb contains the function geoConvert that addresses these questions directly and efficiently.

Using GEOmetadb

Once GEOmetadb is installed (see the Bioconductor website for full installation instructions), we are ready to begin. This package does not come with a pre-installed version of the database. This has the advantage that the user will get the most up-to-date version of the database to start; the database can be re-downloaded using the same command as often as desired. First, load the library.

```
> library(GEOmetadb)
```

The download and uncompress steps are done automatically with a single command, getSQLiteFile.

```
> getSQLiteFile()
Unzipping...
Metadata associate with downloaded file:
                name               value
1     schema version                 1.0
2 creation timestamp 2010-06-19 10:55:24
[1] "/Volumes/TravelDrive/BiocCourses/CSHL/2010/GEOmetadb.sqlite"
```

The default storage location is in the current working directory and the default filename is "GEOmetadb.sqlite"; it is best to leave the name unchanged unless there is a pressing reason to change it. Since this SQLite file is of key importance in GEOmetadb, it is perhaps of some interest to know some details about the file itself.

```
> file.info("GEOmetadb.sqlite")
                       size isdir mode               mtime               ctime
GEOmetadb.sqlite 1228368896 FALSE  644 2010-06-24 07:48:17 2010-06-24 07:48:17
                               atime uid gid  uname grname
GEOmetadb.sqlite 2010-06-24 07:48:18 501  20 sdavis  staff
```

Now, the SQLite file is available for connection. The standard DBI package functionality as implemented in RSQLite function dbConnect makes the connection to the database. The dbDisconnect function disconnects the connection.

```
> con <- dbConnect(SQLite(), "GEOmetadb.sqlite")
> dbDisconnect(con)
[1] TRUE
```

The variable con is an RSQLite connection object.

A word about SQL

The Structured Query Language, or SQL, is a very powerful and standard way of working with relational data. GEO is composed of several data types, all of which are related to each other; in fact, NCBI uses a relational SQL database for metadata storage and querying. SQL databases and SQL itself are designed specifically to work efficiently with just such data. While the goal of many programming projects and programmers is to hide the details of SQL from the user, we are of the opinion that such efforts may be counterproductive, particularly with complex data and the need for textit{ad hoc} queries, both of which are characteristics with GEO metadata. We have taken the view that exposing the power of SQL will enable users to maximally utilize the vast data repository that is GEO. We understand that many users are not accustomed to working with SQL and, therefore, have devoted a large section of the vignette to working examples. Our goal is not to teach SQL, so a quick tutorial of SQL is likely to be beneficial to those who have not used it before. Many such tutorials are available online and can be completed in 30 minutes or less.

GEOmetadb Examples

The functionality covered in this section is covered in much more detail in the DBI and RSQLite package documentation. We cover enough here only to be useful. Again, we connect to the database.

```
> con <- dbConnect(SQLite(), "GEOmetadb.sqlite")
```

The dbListTables function lists all the tables in the SQLite database handled by the connection object con.

```
> geo_tables <- dbListTables(con)
> geo_tables
 [1] "gds"                 "gds_subset"          "geoConvert"
 [4] "geodb_column_desc" "gpl"                 "gse"
 [7] "gse_gpl"             "gse_gsm"             "gsm"
[10] "metaInfo"            "sMatrix"
```

There is also the dbListFields function that can list database fields associated with a table.

```
> dbListFields(con, "gse")
 [1] "ID"                   "title"                "gse"
 [4] "status"               "submission_date"      "last_update_date"
 [7] "pubmed_id"            "summary"              "type"
[10] "contributor"          "web_link"             "overall_design"
[13] "repeats"              "repeats_sample_list"  "variable"
[16] "variable_description" "contact"              "supplementary_file"
```

Sometimes it is useful to get the actual SQL schema associated with a table. As an example of doing this and using an RSQLite shortcut function, sqliteQuickSQL, we can get the table schema for the gpl table.

```
> sqliteQuickSQL(con, "PRAGMA TABLE_INFO(gpl)")
   cid                 name type notnull dflt_value pk
1    0                   ID REAL       0       <NA>  0
2    1                title TEXT       0       <NA>  0
3    2                  gpl TEXT       0       <NA>  0
4    3               status TEXT       0       <NA>  0
5    4      submission_date TEXT       0       <NA>  0
6    5     last_update_date TEXT       0       <NA>  0
7    6           technology TEXT       0       <NA>  0
8    7         distribution TEXT       0       <NA>  0
9    8             organism TEXT       0       <NA>  0
10   9         manufacturer TEXT       0       <NA>  0
11  10 manufacture_protocol TEXT       0       <NA>  0
12  11              coating TEXT       0       <NA>  0
13  12       catalog_number TEXT       0       <NA>  0
14  13              support TEXT       0       <NA>  0
15  14          description TEXT       0       <NA>  0
16  15             web_link TEXT       0       <NA>  0
17  16              contact TEXT       0       <NA>  0
18  17       data_row_count REAL       0       <NA>  0
```

```
19  18   supplementary_file TEXT       0       <NA>   0
20  19         bioc_package TEXT       0       <NA>   0
```

Select 5 records from the gse table and show the first 7 columns.

```
> rs <- dbGetQuery(con, "select * from gse limit 5")
> rs[, 1:7]
  ID                                                 title  gse
1  1                                   NHGRI_Melanoma_class GSE1
2  2                                 Cerebellar development GSE2
3  3              Renal Cell Carcinoma Differential Expression GSE3
4  4      Diurnal and Circadian-Regulated Genes in Arabidopsis GSE4
5  5 Global profile of germline gene expression in C. elegans GSE5
                 status submission_date last_update_date pubmed_id
1 Public on Jan 22 2001      2001-01-22       2005-05-29  10952317
2 Public on Apr 26 2001      2001-04-19       2005-05-29        NA
3 Public on Jul 19 2001      2001-07-19       2005-05-29  11691851
4 Public on Jul 20 2001      2001-07-20       2005-05-29  11158533
5 Public on Jul 24 2001      2001-07-24       2005-07-18  11030340
```

Get the GEO series accession and title from GEO series that were submitted by “Sean Davis”. The “%” sign is used in combination with the “like” operator to do a “wildcard” search for the name “Sean Davis” with any number of characters before or after or between “Sean” and “Davis”.

```
> rs <- dbGetQuery(con, paste("select gse,title from gse where",
+     "contributor like '%Sean%Davis%'", sep = " "))
> rs
        gse
1   GSE2553
2   GSE4406
3   GSE5357
4   GSE7376
5   GSE8486
6   GSE9328
7   GSE7882
8  GSE14543
9   GSE5481
10 GSE18544
11 GSE16087
12 GSE16088
13 GSE16091
14 GSE16102
15 GSE20016

1
2                                            Gene expression profiling of CD4+ T-cells and GM6990 lymp
3
4                                                                     Detection of novel amplification uni
5                                                                     Whole genome DNAse hypersensitivity
6                                                                                               ATF2 
7                                 Gene Expression and Comparative Genomic Hybridization of Ductal Carcinoma
8                                                                                  A molecular function 
9                                                                                               BRAF 
10     Expression Profiling of a Mouse Xenograft Model of "Triple-Negative" Breast Cancer Brain Metas
11                                                                             Gene expression profiles 
12                                                                              Gene expression profiles
13                                                                        Gene expression profiles of hum
```

```
14                                                              Gene expression profiles of canine a
15 Analyses of Human Brain Metastases of Breast Cancer Reveal the Association between HK2 Up-Regulati
```

As another example, GEOmetadb can find all samples on GPL96 (Affymetrix hgu133a) that have .CEL files available for download.

```
> rs <- dbGetQuery(con, paste("select gsm,supplementary_file",
+     "from gsm where gpl='GPL96'", "and supplementary_file like '%CEL.gz'"))
> dim(rs)
[1] 15087     2
```

But why limit to only GPL96? Why not look for all Affymetrix arrays that have .CEL files? And list those with their associated GPL information, as well as the Bioconductor annotation package name?

```
> rs <- dbGetQuery(con, paste("select gpl.bioc_package,gsm.gpl,",
+     "gsm,gsm.supplementary_file", "from gsm join gpl on gsm.gpl=gpl.gpl",
+     "where gpl.manufacturer='Affymetrix'", "and gsm.supplementary_file like '%CEL.gz' "))
> rs[1:5, ]
  bioc_package   gpl    gsm
1       hu6800 GPL80 GSM575
2       hu6800 GPL80 GSM576
3       hu6800 GPL80 GSM577
4       hu6800 GPL80 GSM578
5       hu6800 GPL80 GSM579
                                                                  supplementary_file
1 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM575/GSM575.cel.gz
2 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM576/GSM576.cel.gz
3 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM577/GSM577.cel.gz
4 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM578/GSM578.cel.gz
5 ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSMnnn/GSM579/GSM579.cel.gz
```

Of course, we can combine programming and data access. A simple sapply example shows how to query each of the tables for number of records.

```
> getTableCounts <- function(tableName, conn) {
+     sql <- sprintf("select count(*) from %s", tableName)
+     return(dbGetQuery(conn, sql)[1, 1])
+ }
> do.call(rbind, sapply(geo_tables, getTableCounts, con, simplify = FALSE))
                     [,1]
gds                  2089
gds_subset          11858
geoConvert        2044214
geodb_column_desc     104
gpl                  7648
gse                 17450
gse_gpl             22391
gse_gsm            506551
gsm                460450
metaInfo                2
sMatrix             20498
```

Large-scale consumers of GEO data might want to convert GEO entity type from one to others, e.g. finding all GSM and GSE associated with "GPL96". Function goeConvert does the conversion with a very fast mapping between entity types.

Covert "GPL96" to other possible types in the GEOmetadb.sqlite.

```
> conversion <- geoConvert("GPL96")
```

Check what GEO types and how many entities in each type in the conversion.

```
> lapply(conversion, dim)
$gse
[1] 768   2

$gsm
[1] 24259     2

$gds
[1] 254   2

$sMatrix
[1] 796   2
> conversion$gse[1:5, ]
  from_acc   to_acc
1    GPL96  GSE1000
2    GPL96 GSE10024
3    GPL96 GSE10043
4    GPL96 GSE10072
5    GPL96 GSE10089
> conversion$gsm[1:5, ]
  from_acc    to_acc
1    GPL96 GSM100386
2    GPL96 GSM100454
3    GPL96 GSM100455
4    GPL96 GSM100456
5    GPL96 GSM100457
> conversion$gds[1:5, ]
  from_acc  to_acc
1    GPL96 GDS1023
2    GPL96 GDS1036
3    GPL96 GDS1050
4    GPL96 GDS1062
5    GPL96 GDS1063
> conversion$sMatrix[1:5, ]
  from_acc                        to_acc
1    GPL96  GSE1000_series_matrix.txt.gz
2    GPL96 GSE10024_series_matrix.txt.gz
3    GPL96 GSE10043_series_matrix.txt.gz
4    GPL96 GSE10072_series_matrix.txt.gz
5    GPL96 GSE10089_series_matrix.txt.gz
```

Now, for something a bit more complicated, we would like to find all the human breast cancer-related Affymetrix gene expression GEO series.

```
> sql <- paste("SELECT DISTINCT gse.title,gse.gse", "FROM", "  gsm JOIN gse_gsm ON gsm.gsm=gse_gsm.gs
+     "  JOIN gse ON gse_gsm.gse=gse.gse", "  JOIN gse_gpl ON gse_gpl.gse=gse.gse",
+     "  JOIN gpl ON gse_gpl.gpl=gpl.gpl", "WHERE", "  gsm.molecule_ch1 like '%total RNA%' AND",
+     "  gse.title LIKE '%breast cancer%' AND", "  gpl.organism LIKE '%Homo sapiens%'",
+     sep = " ")
> rs <- dbGetQuery(con, sql)
> dim(rs)
[1] 213   2
> print(rs[1:5, ], right = FALSE)
```

```
  title
1 A Modular Analysis of Breast Cancer Reveals a Novel Low-Grade Molecular Signature in Estrogen Recep
2 A Phase II Study of Neoadjuvant Gemcitabine Plus Doxorubicin Followed by Gemcitabine Plus Cisplatin
3 A Supervised Risk Predictor of Breast Cancer Based on Biological Subtypes
4 A functional and regulatory network associated with PIP expression in human breast cancer
5 A gene expression signature identifies two prognostic subgroups of basal breast cancer
  gse
1 GSE2294
2 GSE8465
3 GSE10886
4 GSE11627
5 GSE21653
```

Finally, it is probably a good idea to close the connection, please see DBI for detail.

```
> dbDisconnect(con)
[1] TRUE
```

If you want to remove old GEOmetadb.sqlite file before retrieve a new version from the server, execute the following code:

```
> file.remove("GEOmetadb.sqlite")
[1] TRUE
```

CHAPTER

TWO

ILLUMINA METHYLATION DATA ANALYSIS

Author Sean Davis

Contact sdavis2@mail.nih.gov

Date 2010-06-28

2.1 Introductory Remarks

Gene expression patterns are very important in understanding any biologic system. The regulation of gene expression is a complex, poorly understood process. However, DNA methylation is one biological process that is known to be important in gene regulation. In short, DNA methylation is a chemical modification of DNA CpG sites in which a methyl group is added number 5 carbon of the cytosine pyrimidine ring. In humans, the DNA methyltransferases (DNMT1, DNMT3a, and DNMT3b) are the enzymes responsible for carrying out the methylation.

2.1.1 The Illumina Methylation Platform

The Illumina GoldenGate methylation profiling technology specifically targets more than 1500 CpG sites throughout the genome, specifically targeting approximately 700 “cancer genes”. Samples are run in 96-well format, making the technology very high-throughput. After a two-color hybridization, a laser captures the intensities associated with the methylated state and the accompanying unmethylated state. The Illumina BeadStudio software is then used for quality control and basic visualization tasks. A newer platform, the Illumina Infinium Methylation platform provides a more “whole-genome” view of DNA methylation. Utilizing the Infinium profiling technology on bisulfite-treated DNA, the methylation status of more than 25,000 individual CpG sites is assayed simultaneously. This package can handle both types of data.

2.2 Illumina Methylation Data Analysis

The methylumi package provides convenient mechanisms for loading the results of the Illumina methylation platform into R/Bioconductor. Classes based on common Bioconductor classes for encapsulating the data and facilitate data manipulation are at the core of the package, with methods for quality control, normalization, and plotting.

2.2.1 Overview of the Golden Gate Methylation Array

The GGHumanMethCancerPanel.db package is a standard Bioconductor annotation package but contains a few other pieces of annotation of interest.

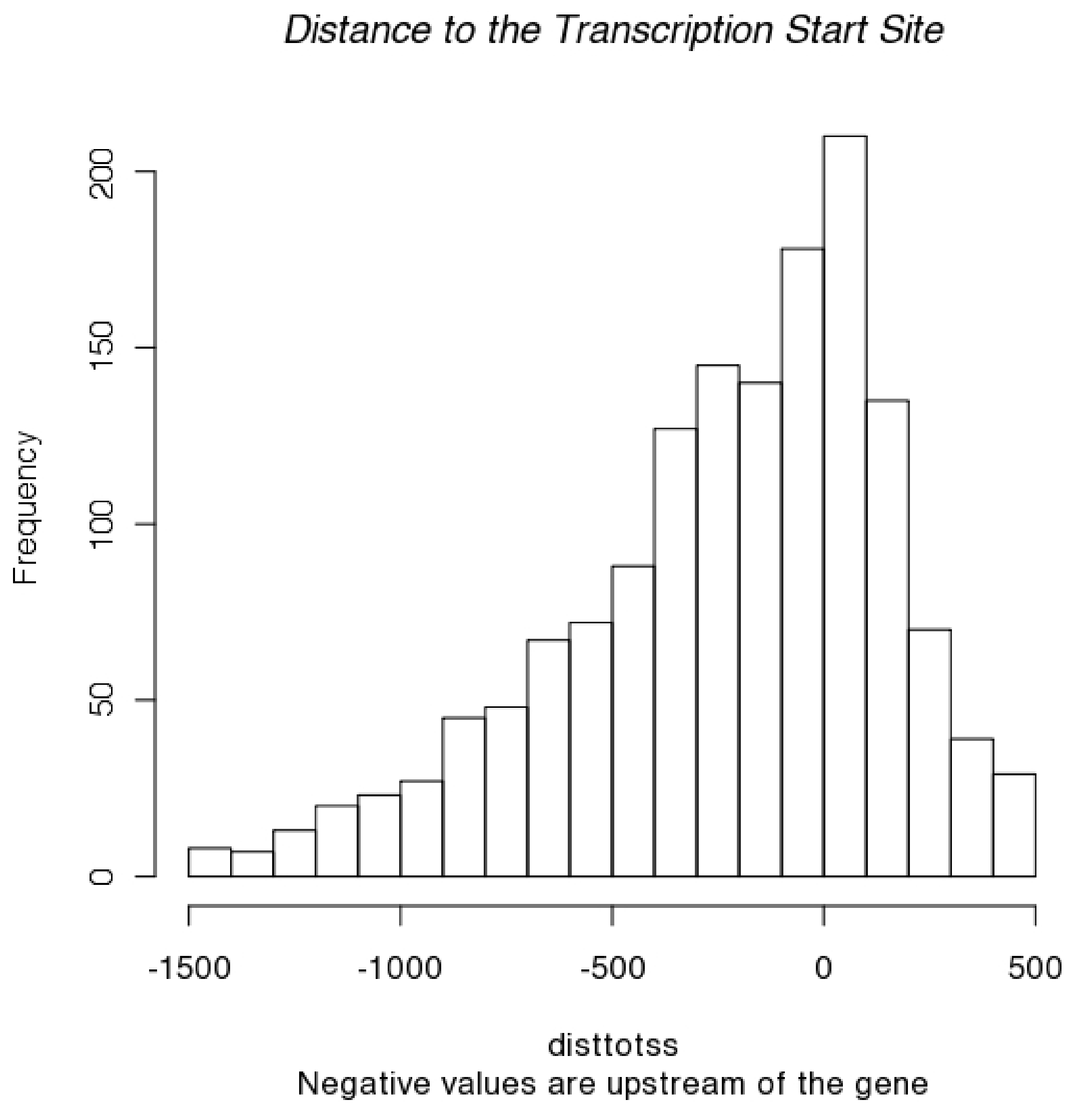
```
> library(GGHumanMethCancerPanelv1.db)
```

A parameter of interest is the proportion of CpG sites being assayed that are actually in a CpG island (a stretch of high GC content) as defined by Illumina.

```
> cpgisland = toTable(GGHumanMethCancerPanelv1ISCPGISLAND)[,
+     2] == 1
> table(cpgisland)
cpgisland
FALSE  TRUE
  461  1044
```

Another is the location relative to the transcription start site (TSS). This information is also accessible from the annotation package.

```
> disttotss = toTable(GGHumanMethCancerPanelv1DISTTOTSS)[,
+     2]
> hist(disttotss, breaks = "scott", main = "Distance to the Transcription Start Site",
+     sub = "Negative values are upstream of the gene")
```



2.2.2 A Simple Example

This example borrows heavily from the methylumi vignette. This example was part of a beta-test of the Illumina methylation platform in collaboration with Illumina. The data are extracted from a larger study examining methylation in hematologic tissues, including malignancies (unrelated to the later example). The experimental question is rather contrived:

> Are there methylation differences between male and female samples?

One would hope that this will be possible given the knowledge that the female *X* chromosome undergoes so-called “X-inactivation”, a process that leads to methylation of one entire copy of the *X* chromosome in females.

Loading data

After exporting the data from BeadStudio, methylumi can read them in with a single command. To include rich sample annotation, it is possible to supply a data.frame including that sample annotation. This can be read from disk or constructed on-the-fly for flexibility. If used, a SampleID column must be present and match the sample IDs used in the BeadStudio export file. Also, if a column called SampleLabel is present in the data frame and it includes unique names, the values from that column will be used for the sampleNames of the resulting MethyLumiSet.

Two different formats can be read by methylumi. The *Final Report* format includes all the data in a single file. The package will look **[Header]** in the file to determine when that file format is to be used. The data block **[Sample Methylation Profile]** needs to be present in the *Final Report* format. If the data block **[Control Probe Profile]** is present, these data will be included in the QCdata of the resulting MethyLumiSet object. The second format can be a simple tab-delimited text file with column headers from BeadStudio. If this format is used, the sample data and the QC data can be in separate files. The QC data need not be present for either format, but it can be helpful for quality control of data. For the examples in this vignette, a small sample set run on the Illumina GoldenGate platform will be used and the file format is the tab-delimited format.

```
> library(methylumi)
> samps <- read.table(system.file("extdata/samples.txt",
+     package = "methylumi"), sep = "\t",
+     header = TRUE)
> mldat <- methylumiR(system.file("extdata/exampledata.samples.txt",
+     package = "methylumi"), qcfile = system.file("extdata/exampledata.controls.txt",
+     package = "methylumi"), sampleDescriptions = samps)
```

The table below gives the sample annotation information as specified by the experimenter.

Sample information for the example data set

SampleID	SampleLabel	Gender
1632405013_R006_C012	M_1	M
1632405013_R007_C001	M_2	M
1632405013_R007_C002	M_3	M
1632405013_R007_C003	M_4	M
1632405013_R007_C008	F_5	F
1632405013_R007_C009	F_6	F
1632405013_R007_C010	F_7	F
1632405013_R007_C011	F_8	F
1632405013_R007_C012	M_9	M
1632405013_R008_C001	F_10	F

The resulting data structure, the *MethyLumiSet*, looks like this:

```
> showClass("MethyLumiSet")
Class "MethyLumiSet" [package "methylumi"]

Slots:

Name:                    QC           history
Class:         QCDataOrNULL        data.frame

Name:             assayData         phenoData
Class:            AssayData AnnotatedDataFrame

Name:           featureData    experimentData
Class: AnnotatedDataFrame             MIAME

Name:            annotation      protocolData
Class:            character AnnotatedDataFrame

Name:     .__classVersion__
Class:             Versions
```

```
Extends:
Class "MethyLumi", directly
Class "eSet", by class "MethyLumi", distance 2
Class "VersionedBiobase", by class "MethyLumi", distance 3
Class "Versioned", by class "MethyLumi", distance 4
```

Only a subset of an entire plate is included here for illustration purposes. The mldat object now contains the data (in an eSet-like object) and quality control information (available as `QCdata(mldat)`, also an eSet-like object) from a set of experiments. The details of what was loaded can be viewed:

```
> mldat
Object Information:
MethyLumiSet (storageMode: environment)
assayData: 1536 features, 10 samples
  element names: Avg_NBEADS, BEAD_STDERR, betas, methylated, pvals, unmethylated
protocolData: none
phenoData
  sampleNames: M_1, M_2, ..., F_10  (10 total)
  varLabels and varMetadata description:
    sampleID: sampleID
    SampleLabel: SampleLabel
    Sample: Sample
    Gender: Gender
featureData
  featureNames: AATK_E63_R, AATK_P519_R, ..., ZP3_
  P220_F  (1536 total)
  fvarLabels and fvarMetadata description:
    TargetID: NA
    ProbeID: NA
    ...: ...
    PRODUCT: NA
    (17 total)
experimentData: use 'experimentData(object)'
Annotation:
Major Operation History:
            submitted            finished
1 2010-06-23 17:23:21 2010-06-23 17:23:22

1 methylumiR(filename = system.file("extdata/exampledata.samples.txt",     package = "methylumi"), q
```

Accessors for various slots of the resulting object are outlined in the online help. It is worth noting, though, that the QCdata method will return another eSet-like object of class MethyLumiQC; the data contained here can be useful if an array has failed.

Note that the assayData names have been changed from the original column identifiers in the data file from Illumina. The mappings are available via the function getAssayDataNameSubstitutions().

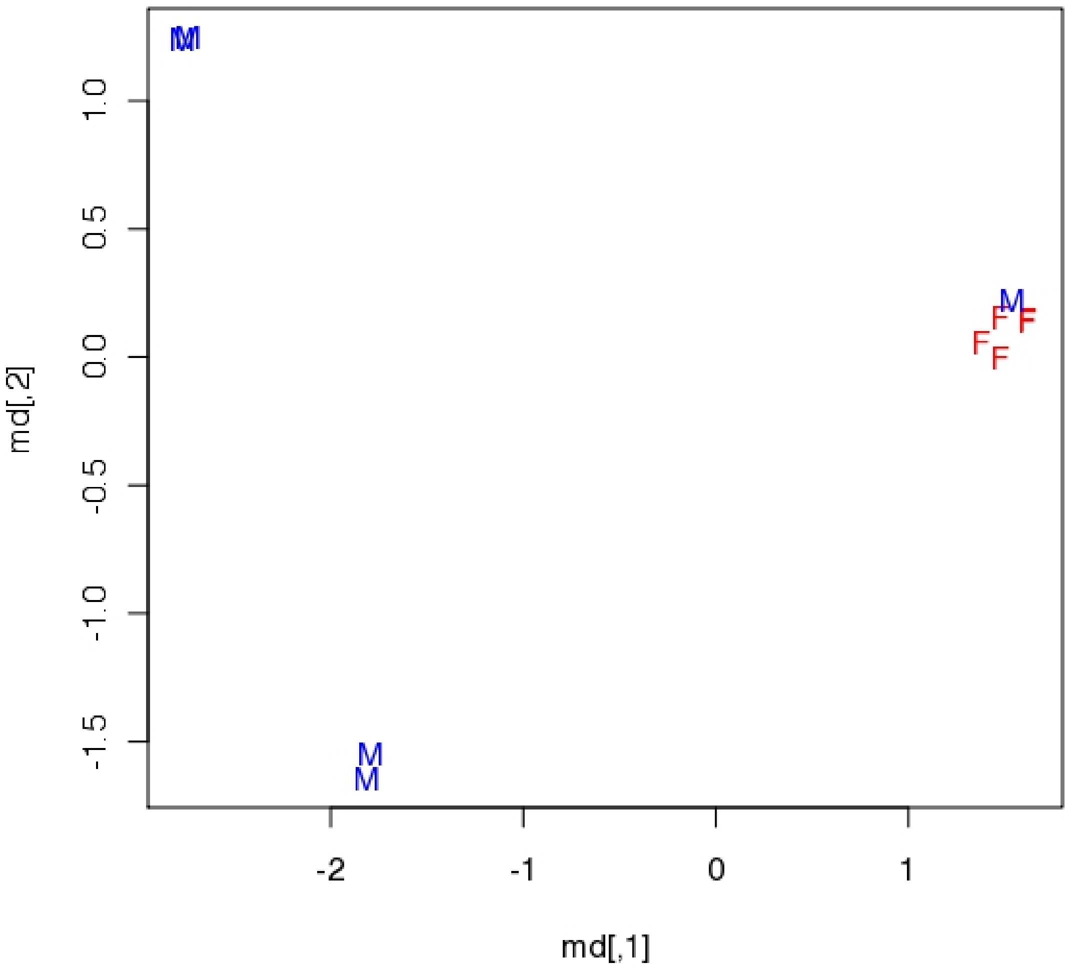
```
> getAssayDataNameSubstitutions()
           regex       result
1       AVG_Beta        betas
2 Detection Pval        pvals
3     Signal CY3 unmethylated
4     Signal CY5   methylated
5     Signal_Red unmethylated
6     Signal_Grn   methylated
7       Signal_A unmethylated
8       Signal_B   methylated
```

Quality Control

The data that are included with the methylumi package are all normal samples from the same tissue. The samples are labeled with the presumed gender. The data are meant to be illustrative of some typical quality-control and sample-labeling problems. In order to get a quick overview of the samples, it is useful to look at a multidimensional scaling (MDS) plot of the samples, using only probes on the X chromosome. Since females undergo X-inactivation, they should show something approximating hemi-methylation on that chromosome while males should show very little methylation on the X chromosome.

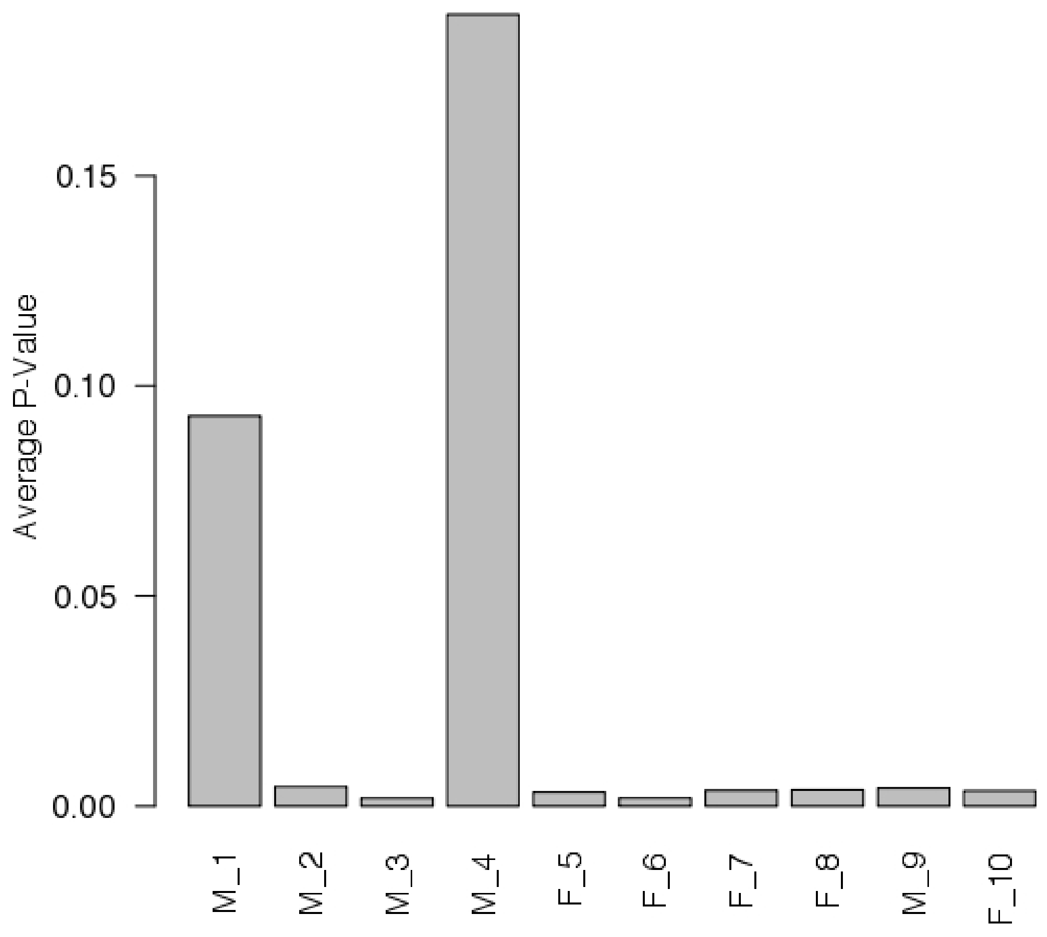
```
> md <- cmdscale(dist(t(exprs(mldat)[fData(mldat)$CHROMOSOME ==
+     "X", ])), 2)
```

```
> plot(md, pch = c("F", "M")[pData(mldat)$Gender],
+     col = c("red", "blue")[pData(mldat)$Gender])
```



The MDS plot shows that the males and females are quite distinct except for a single male that groups with the females. Upon consultation with the laboratory investigator, the sample was found to be mislabeled. Also, it is worth noting that the males do not cluster nearly as tightly as the females. A quick evaluation of the p-values for detection (as reported by the BeadStudio software) for the samples will show what the problem is:

```
> avgPval <- colMeans(pvals(mldat))
> par(las = 2)
> barplot(avgPval, ylab = "Average P-Value")
```



So, it is quite obvious that there are two arrays that fail QC with a large percentage of the reporters showing lack of measurement.

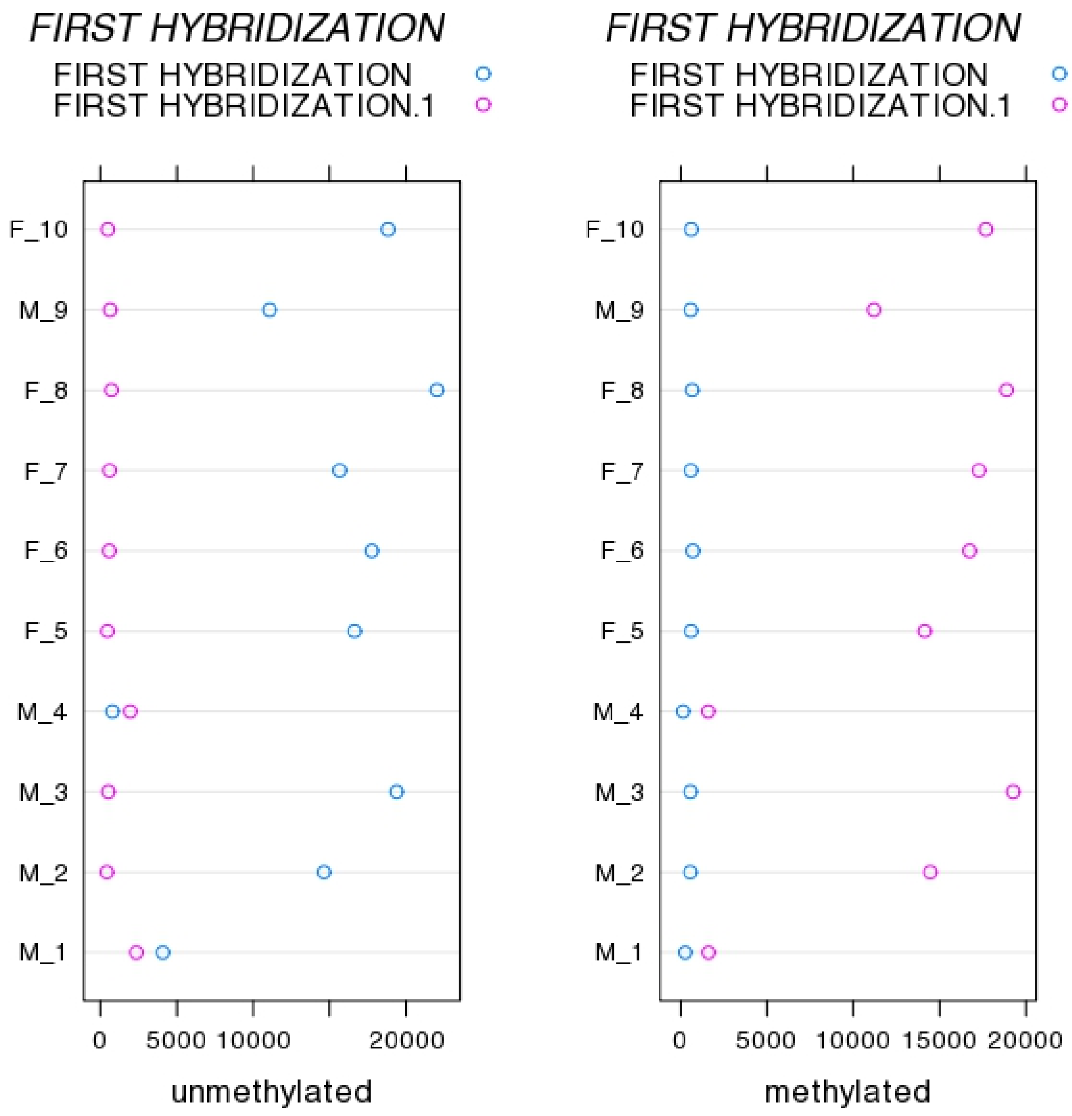
It is also possible to use the qcplot method in combination with the controlTypes method to examine the QC data in more detail. The control types for the GoldenGate platform are:

```
> controlTypes(mldat)
[1] "ALLELE SPECIFIC EXTENSION"
[2] "EXTENSION GAP"
[3] "FIRST HYBRIDIZATION"
[4] "GENDER"
```

```
[5] "NEGATIVE"
[6] "PCR CONTAMINATION"
[7] "SECOND HYBRIDIZATION"
```

Looking more closely at the hybridization controls (`FIRST HYBRIDIZATION`) is telling here:

```
> qcplot(mldat, "FIRST HYBRIDIZATION")
```



So, it appears that the hybridization controls (at least) failed for samples `M_1` and `M_4`, which might help explain why the samples failed. In any case, a trip back to the experimenter is probably warranted for these two samples.

Normalization

The Illumina platform shows a significant dye bias in the two channels which will lead to bias in the estimates of Beta. Therefore, some normalization is required. The function, normalizeMethyLumiSet, does this normalization. Basically, it looks at the median intensities in the methylated and unmethylated channels at very low and very high beta values and sets these medians equal. Using the transformed unmethylated and methylated values, new beta values are calculated using one of two `map` functions. The ratio map function is the default and is the same as used by Illumina in the BeadStudio software, but values using the atan map function selection should be similar.

First, a bit of cleanup is needed. The two samples with significantly poorer quality are removed. The gender of the mis-labeled sample is also corrected.

```
> toKeep <- (avgPval < 0.05)
> pData(mldat)$Gender[9] <- "F"
> mldat.norm <- normalizeMethyLumiSet(mldat[,
+     toKeep])
```

Example Analysis

As a simple example of an analysis, we can look for the differences between males and females. We already know there is a strong difference based on simple unsupervised methods (the MDS plot). However, methylation is particularly informative for sex differences because females undergo X-inactivation and are, therefore, expected to have one copy of the X chromosome largely methylated. *Note that, while limma is used here to illustrate a point, an appropriate statistical framework for finding differential methylation targets based on the Illumina methylation platforms with data that is not normally distributed under the null is a current research topic for a number of groups.*

```
> library(limma)
> dm <- model.matrix(~1 + Gender, data = pData(mldat.norm))
> colnames(dm)
[1] "(Intercept)" "GenderM"
> fit1 <- lmFit(exprs(mldat.norm), dm)
> fit2 <- eBayes(fit1)
> tt <- topTable(fit2, coef = 2, genelist = fData(mldat.norm)[,
+     c("SYMBOL", "CHROMOSOME")], number = 1536)
> x <- aggregate(tt$adj.P.Val, by = list(tt$CHROMOSOME),
+     median)
> colnames(x) <- c("Chromosome", "Median adjusted P-value")
```

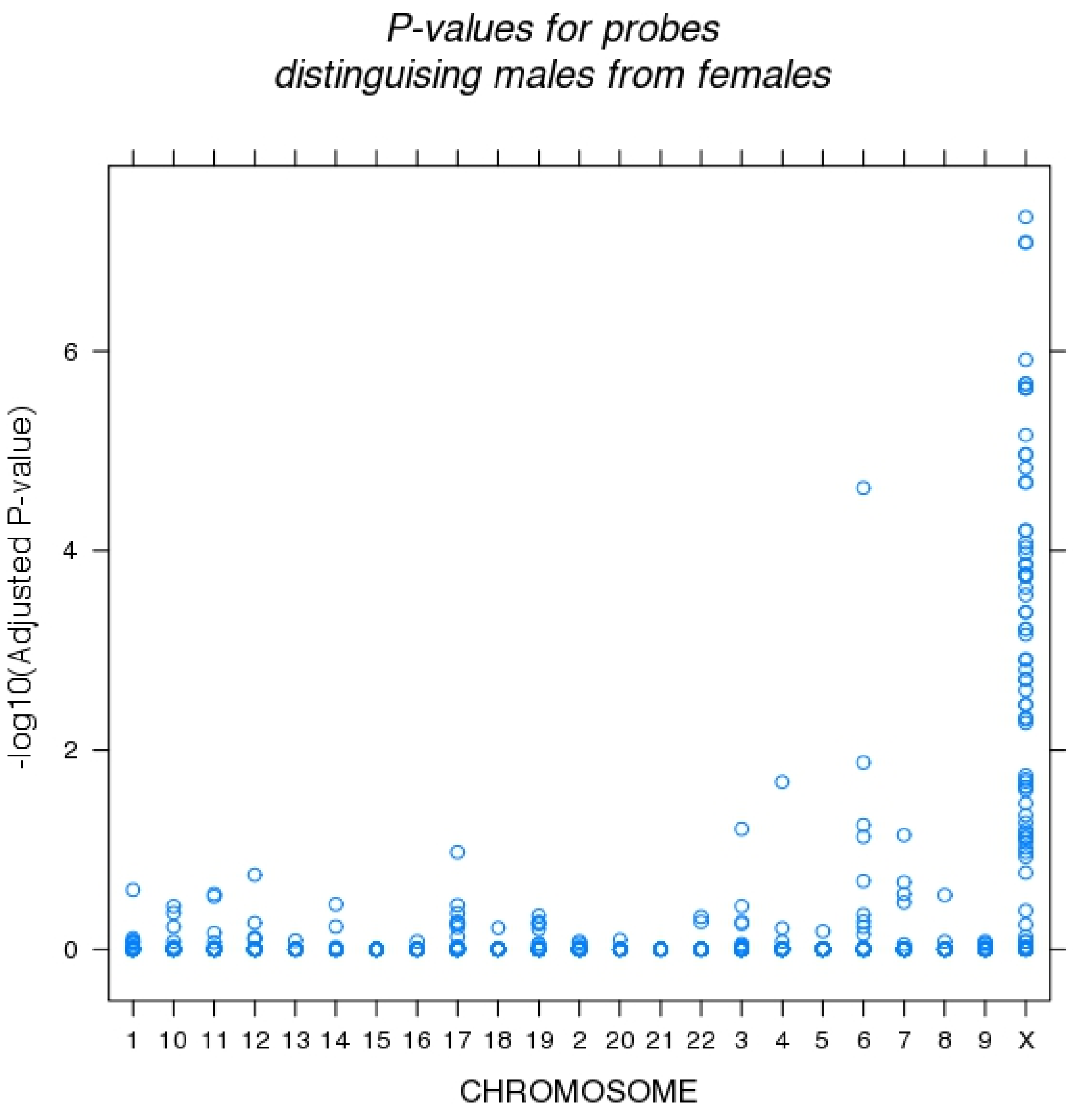
```
> xt = ascii(x, caption = "The median adjusted p-value for each chromosome showing that the X-chromos
+     caption.level = "e", include.rownames = FALSE)
> print(xt)
```

The median adjusted p-value for each chromosome showing that the X-chromosome is highly significantly different between males and females

Chromosome	Median adjusted P-value
1	1.00
10	1.00
11	1.00
12	1.00
13	1.00
14	1.00
15	1.00
16	1.00
17	1.00
18	1.00
19	1.00
2	1.00
20	1.00
21	1.00
22	1.00
3	1.00
4	1.00
5	1.00
6	1.00
7	1.00
8	1.00
9	1.00
X	0.00

Looking at the median adjusted p-values for the resulting differences (calculated using limma), one can quickly see that the X chromosome is, indeed, quite significantly different, on the whole, between males and females. The actual p-values are plotted to show the distribution.

```
> library(lattice)
> print(xyplot(-log10(adj.P.Val) ~ CHROMOSOME,
+     tt, ylab = "-log10(Adjusted P-value)",
+     main = "P-values for probes\ndistinguising males from females"))
```



2.2.3 Methylation Profiles of Breast Cancer

To get a better feeling for the complexity in methylation profiles, it is instructive to look at a larger data set. The data set of interest is taken from Holm et al [1]. Their abstract explains the work at a high level:

> *Introduction*: Five different molecular subtypes of breast cancer have been identified through gene expression profiling. Each subtype has a characteristic expression pattern suggested to partly depend on cellular origin. We aimed to investigate whether the molecular subtypes also display distinct methylation profiles.
>
> *Methods*: We analysed methylation status of 807 cancer-related genes in 189 fresh frozen primary breast tumours and four normal breast tissue samples using an array-based methylation assay.
>
> *Results*: Unsupervised analysis revealed three groups of breast cancer with characteristic methylation patterns. The three groups were associated with the luminal A, luminal B and basal-like molecular subtypes of breast cancer, respectively, whereas cancers of the HER2-enriched and normal-like subtypes were

[1] Holm K, Hegardt C, Staaf J, Vallon-Christersson J et al. Molecular subtypes of breast cancer are associated with characteristic DNA methylation patterns. Breast Cancer Res 2010 Jun 18;12(3):R36. PMID: 20565864

> distributed among the three groups. The methylation frequencies were significantly different between subtypes, with luminal B and basal-like tumours being most and least frequently methylated, respectively. Moreover, targets of the polycomb repressor complex in breast cancer and embryonic stem cells were more methylated in luminal B tumours than in other tumours. BRCA2-mutated tumours had a particularly high degree of methylation. Finally, by utilizing gene expression data, we observed that a large fraction of genes reported as having subtype-specific expression patterns might be regulated through methylation.
>
> *Conclusions*: We have found that breast cancers of the basal-like, luminal A and luminal B molecular subtypes harbour specific methylation profiles. Our results suggest that methylation may play an important role in the development of breast cancers.

In this little exercise, we will access the data from the NCBI GEO database and then do some simple unsupervised and supervised analyses. The goal is the get a feel for the information content of the data. Note that these data do not include quality control probes, so quality assessment will necessarily be limited. To that end, we will assume that the data are relatively clean (BAD ASSUMPTION) and proceed with the data as deposited in GEO.

```
> library(GEOquery)
> holmdat = getGEO("GSE22210")[[1]]
Found 1 file(s)
GSE22210_series_matrix.txt.gz
File stored at:
/var/folders/zz/zzzivhrRnAmviuee+++b3E++7lI/-Tmp-//RtmpyBY4MD/GPL9183.soft
> holmdat
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1452 features, 193 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: GSM552714, GSM552715, ..., GSM55290
  6  (193 total)
  varLabels and varMetadata description:
    title: NA
    geo_accession: NA
    ...: ...
    data_row_count: NA
    (45 total)
featureData
  featureNames: AATK_E63_R, AATK_P709_R, ..., ZP3_
  P220_F  (1452 total)
  fvarLabels and fvarMetadata description:
    ID: NA
    Gid: NA
    ...: ...
    cg_no: NA
    (16 total)
  additional fvarMetadata: Column, Description
experimentData: use 'experimentData(object)'
Annotation: GPL9183
```

The number of samples and other details of the data set are present in the holmdat object of class ExpressionSet. Note since we are using the beta values directly from GEO, I have left this as an ExpressionSet and not converted to a MethyLumiSet. For the purposes of this exercise, that is not an important step.

Looking closer at the available clinical information provides a high-level view of potential questions:

```
> head(pData(holmdat)[, grep("characteristics",
+     varLabels(holmdat))])
           characteristics_ch1
```

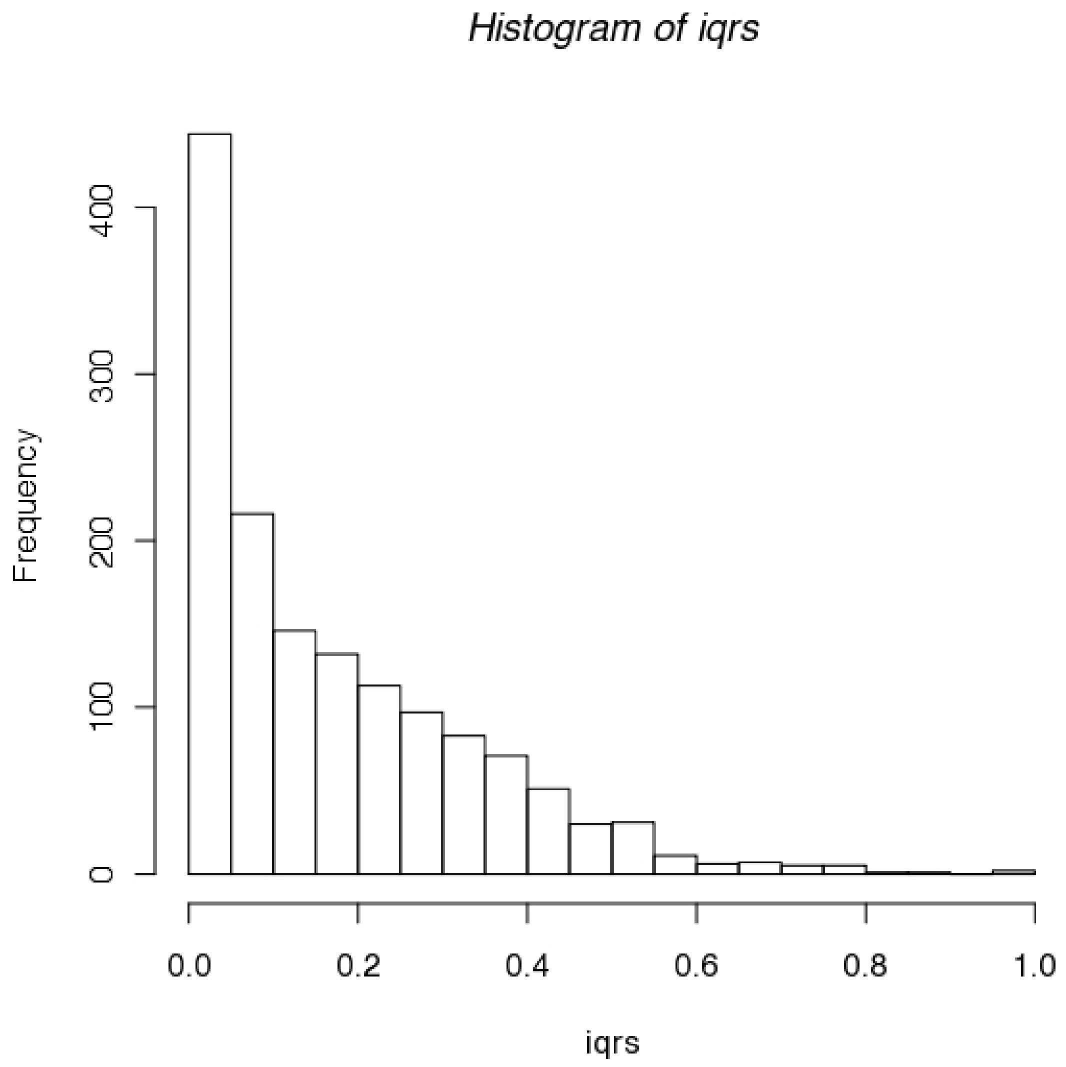
```
GSM552714 tissue: breast tumor
GSM552715 tissue: breast tumor
GSM552716 tissue: breast tumor
GSM552717 tissue: breast tumor
GSM552718 tissue: breast tumor
GSM552719 tissue: breast tumor
            characteristics_ch1.1
GSM552714    family status: brca1
GSM552715 family status: sporadic
GSM552716 family status: sporadic
GSM552717 family status: sporadic
GSM552718 family status: sporadic
GSM552719    family status: brca1
          characteristics_ch1.2
GSM552714           er: er_neg
GSM552715           er: er_neg
GSM552716           er: er_neg
GSM552717           er: er_neg
GSM552718           er: er_neg
GSM552719           er: er_neg
          characteristics_ch1.3
GSM552714         pgr: pgr_neg
GSM552715         pgr: pgr_neg
GSM552716         pgr: pgr_neg
GSM552717         pgr: pgr_neg
GSM552718         pgr: pgr_neg
GSM552719         pgr: pgr_neg
                characteristics_ch1.4
GSM552714  overall survival time: 486
GSM552715 overall survival time: 4378
GSM552716 overall survival time: 5076
GSM552717   overall survival time: NA
GSM552718 overall survival time: 4811
GSM552719  overall survival time: 969
               characteristics_ch1.5
GSM552714  overall survival event: 1
GSM552715  overall survival event: 1
GSM552716  overall survival event: 0
GSM552717 overall survival event: NA
GSM552718  overall survival event: 0
GSM552719  overall survival event: 1
          characteristics_ch1.6
GSM552714     hu subtype: Basal
GSM552715     hu subtype: Basal
GSM552716     hu subtype: Basal
GSM552717     hu subtype: Basal
GSM552718     hu subtype: Basal
GSM552719     hu subtype: Basal
          characteristics_ch1.7
GSM552714         spfpercent: 5
GSM552715         spfpercent: 9
GSM552716        spfpercent: NA
GSM552717        spfpercent: 11
GSM552718        spfpercent: NA
GSM552719        spfpercent: 15
          characteristics_ch1.8
GSM552714        nodestatus: NA
GSM552715  nodestatus: node_neg
```

```
GSM552716  nodestatus: node_neg
GSM552717  nodestatus: node_neg
GSM552718  nodestatus: node_neg
GSM552719  nodestatus: node_neg
          characteristics_ch1.9
GSM552714              grade: 3
GSM552715              grade: 3
GSM552716              grade: 3
GSM552717              grade: 3
GSM552718              grade: 3
GSM552719              grade: 3
          characteristics_ch1.10
GSM552714          hc_cluster: 3
GSM552715          hc_cluster: 3
GSM552716          hc_cluster: 3
GSM552717          hc_cluster: 3
GSM552718          hc_cluster: 3
GSM552719          hc_cluster: 3
          characteristics_ch1.11
GSM552714              fga: 0.63
GSM552715              fga: 0.51
GSM552716              fga: 0.51
GSM552717                fga: NA
GSM552718              fga: 0.39
GSM552719              fga: 0.67
          characteristics_ch1.12
GSM552714        ezh2_gain: Gain
GSM552715        ezh2_gain: Gain
GSM552716      ezh2_gain: NoGain
GSM552717          ezh2_gain: NA
GSM552718        ezh2_gain: Gain
GSM552719        ezh2_gain: Gain
          characteristics_ch1.13
GSM552714             sizemm: NA
GSM552715             sizemm: 30
GSM552716             sizemm: 30
GSM552717             sizemm: 37
GSM552718             sizemm: 22
GSM552719             sizemm: 40
          characteristics_ch1.14
GSM552714         sizestatus: NA
GSM552715          sizestatus: 2
GSM552716          sizestatus: 2
GSM552717          sizestatus: 2
GSM552718          sizestatus: 2
GSM552719          sizestatus: 2
          characteristics_ch1.15
GSM552714            ageyear: 44
GSM552715            ageyear: 81
GSM552716            ageyear: 49
GSM552717            ageyear: 39
GSM552718            ageyear: 49
GSM552719            ageyear: 51
          characteristics_ch1.16
GSM552714           agestatus: 1
GSM552715           agestatus: 2
GSM552716           agestatus: 1
GSM552717           agestatus: 1
```

```
GSM552718           agestatus: 1
GSM552719           agestatus: 2
```

While the phenoData are a bit unwieldy in this form, they are adequate for getting the gist of the data. Let us first remove less informative probes. Using a nonparametric measure of variation is probably warranted given the underlying data that are not normally-distributed (or even bell-shaped). The interquartile range (IQR) fits the bill.

```
> iqrs = apply(exprs(holmdat), 1, IQR)
> summary(iqrs)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00000 0.03507 0.12170 0.17140 0.26630 0.99370
> hist(iqrs, breaks = "scott")
```

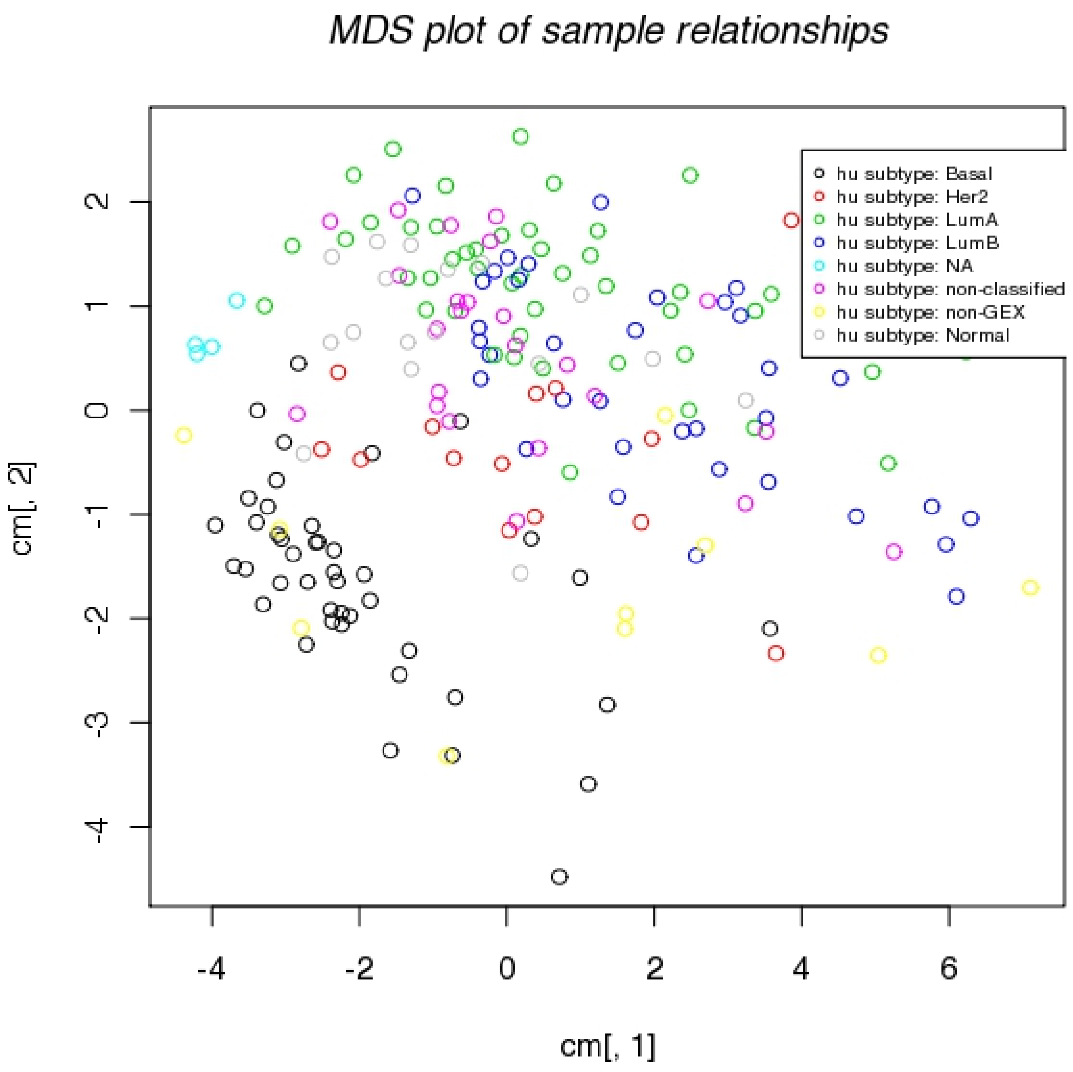


Using an IQR threshold greater than the median is one possible choice for removing probes that are less informative.

```
> holmdatfilt = holmdat[iqrs > median(iqrs),
+     ]
```

A quick unsupervised view of the data is often useful. We could certainly use a hierarchical clustering to do so. However, there are a lot of samples. Therefore, we can try using multidimensional scaling in two dimensions as an alternative. The distance metric used will, of course, affect the results. In this case, simple Euclidean distance can be used. Experiment with others as it interests you.

```
> cm = cmdscale(dist(t(exprs(holmdatfilt))),
+     2)
> plot(cm[, 1], cm[, 2], col = as.factor(pData(holmdatfilt)[,
+     "characteristics_ch1.6"]), main = "MDS plot of sample relationships")
> legend(4, 2.5, levels(as.factor(pData(holmdatfilt)[,
+     "characteristics_ch1.6"])), col = as.factor(levels(as.factor(pData(holmdatfilt)[,
+     "characteristics_ch1.6"]))), pch = 1,
+     cex = 0.6)
```



Note the sample relationships that, with some imagination, are implied by the plot. The Basal subtype is distinct from other subtypes. The Her2 subtype is spatially "between" Basal and the Luminal subtypes. The Luminal subtypes overlap significantly, but there is still some hint that the two "clouds" representing Luminal A and B have different centers. Thus, the biology originally described in gene expression studies appears to be recapitulated in the methylation data, at least at this high level.

Features that differentiate one subtype from another could be useful to help elucidate biological mechanisms of disease or as biomarkers. Using standard parametric methods such as limma may be problematic because of the underlying distribution of the data. Therefore, we might try a different approach and use a classification technique such as random forests to classify the samples. One of the outputs from random forest analysis (at least as implemented in R) is an importance measure for each probe, both overall and for classifying an individual subtype.

When using classification methods for feature selection, it is often important to have approximately "balanced" classes. In this case, we are mainly interested in the three largest breast cancer subtypes, Basal, Luminal A, and Luminal B. So, we can subset the dataset again to include only those samples.

```
> subtypeVector = as.character(pData(holmdatfilt)[,
+     "characteristics_ch1.6"])
> subtypeVector = sub("hu subtype: ", "",
+     subtypeVector)
> cancerIdx = subtypeVector == "Basal" |
+     subtypeVector == "LumA" | subtypeVector ==
+     "LumB"
> holmCancer = holmdatfilt[, cancerIdx]
> pData(holmCancer)$subtype = subtypeVector[cancerIdx]
> table(pData(holmCancer)$subtype)
Basal  LumA  LumB
   43    46    35
```

Now, there are three subtypes represented and, while not perfectly, they are balanced. We move ahead with the random forest analysis of these data.

```
> library(randomForest)
> datamat = t(exprs(holmCancer))
> respvec = factor(pData(holmCancer)$subtype)
```

Actually running the random forest is fairly straightforward. The main parameter to manipulate is the ntree parameter to guarantee that each sample is out-of-bag predicted a few times.

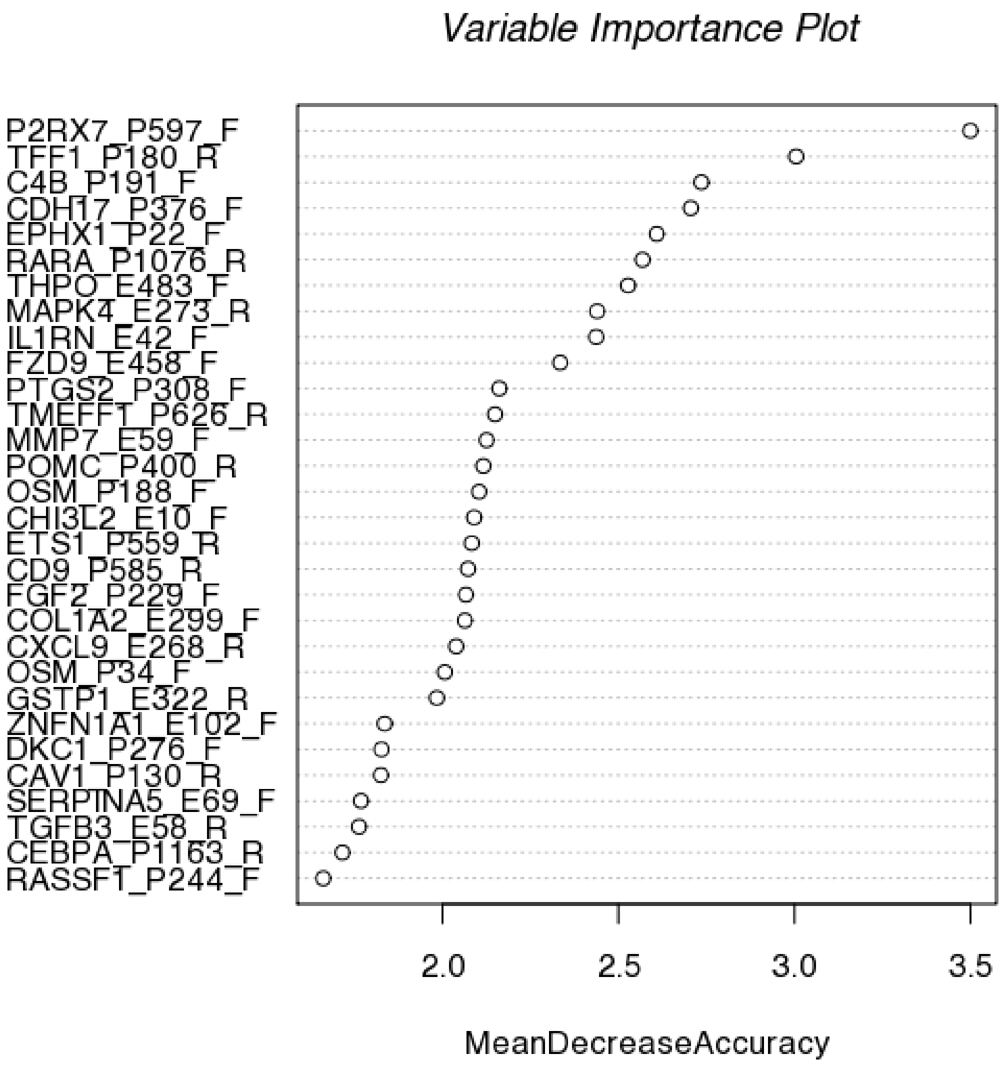
```
> set.seed(1234)
> rfout = randomForest(x = datamat, y = as.factor(respvec),
+     ntree = ncol(datamat) * 10, importance = TRUE,
+     do.trace = 1000)
ntree      OOB      1      2      3
 1000:  21.77%  9.30% 19.57% 40.00%
 2000:  20.97%  6.98% 17.39% 42.86%
 3000:  19.35%  6.98% 17.39% 37.14%
 4000:  20.16%  9.30% 17.39% 37.14%
 5000:  20.97%  9.30% 17.39% 40.00%
 6000:  20.97%  9.30% 17.39% 40.00%
 7000:  20.97%  9.30% 17.39% 40.00%
> rfout
Call:
 randomForest(x = datamat, y = as.factor(respvec), ntree = ncol(datamat) *       10, importance = TRUE
               Type of random forest: classification
                     Number of trees: 7260
No. of variables tried at each split: 26

        OOB estimate of  error rate: 20.97%
Confusion matrix:
      Basal LumA LumB class.error
Basal    39    1    3  0.09302326
```

```
LumA      1   38    7  0.17391304
LumB      0   14   21  0.40000000
```

One can quickly see from the confusion matrix that the ability to distinguish Luminal A from Luminal B is more difficult than distinguishing either from the Basal subtype. Now, the variable importance information (feature selection) is contained in the rfout object. The randomForest package contains a function, varImpPlot that allows a quick visualization of the variable importance measures for the top probes.

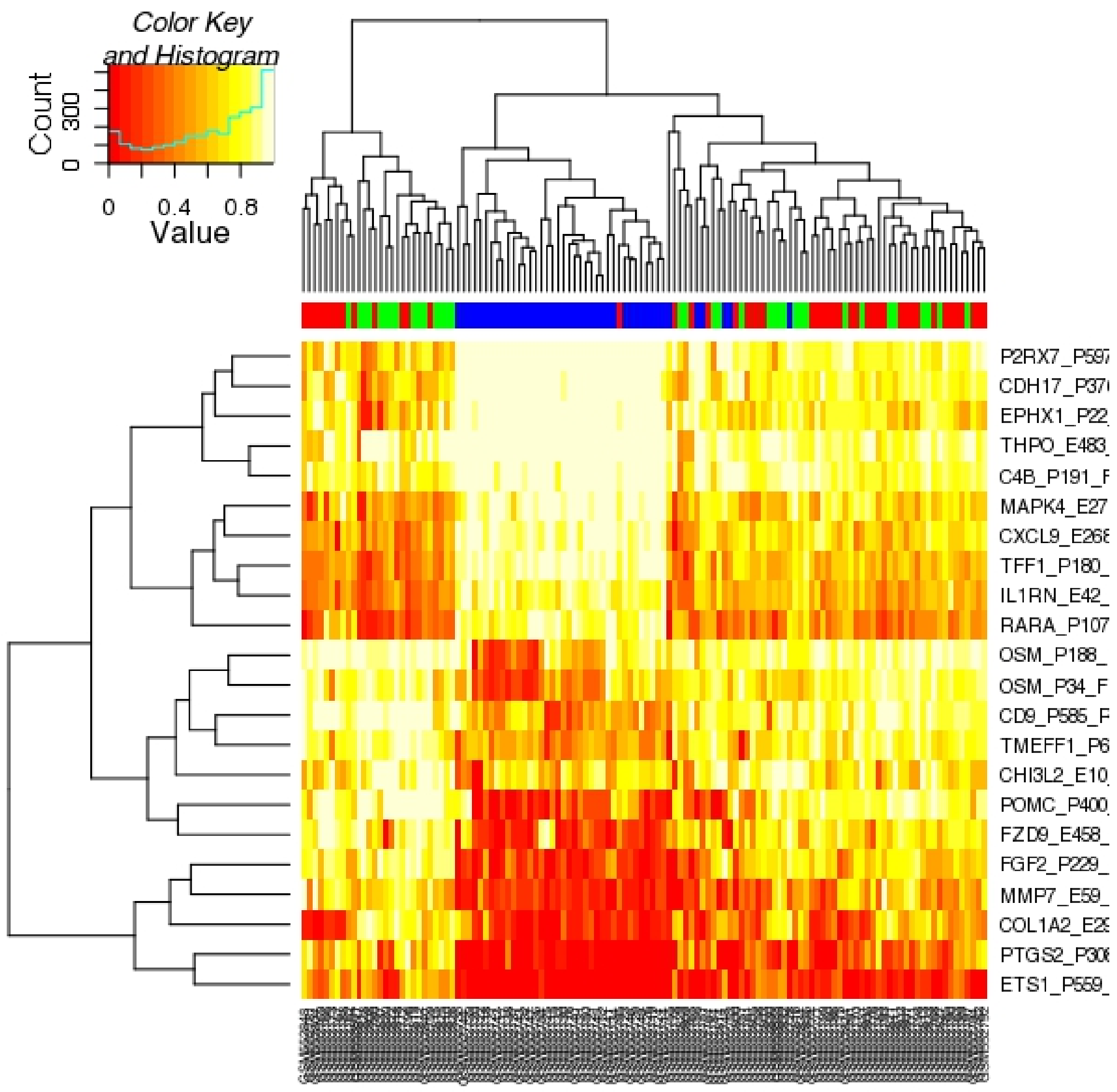
```
> varImpPlot(rfout, type = 1, main = "Variable Importance Plot")
```



To visualize the data on a sample and gene level, we can use a simple heatmap of the top features.

```
> library(gplots)
gdata: read.xls support for 'XLS' (Excel
gdata: 97-2004) files ENABLED.

gdata: read.xls support for 'XLSX' (Excel
gdata: 2007+) files ENABLED.
> topProbes = which(importance(rfout, type = 1) >
+     2)
> heatmap.2(exprs(holmCancer)[topProbes,
+     ], trace = "none", ColSideColors = c("blue",
+     "red", "green")[respvec])
```



In the heatmap, the color bar shows the original sample labels for subtype. The Basal subtype is in blue, luminal A is in red, and Luminal B is in green. As both our confusion matrix and our unsupervised overview of the data suggested, we again see that the distinction between Luminal A and Luminal B is subtle as compared to the Basal subtype.

We could spend quite some time going through individual genes of interest at this point. Gene set enrichment and some type of pathway analysis might be of interest, also. On a more general note, there are many questions that could be addressed with a largish data set like this. What is the effect of a probe being in versus not in a CpG island? What is the relationship between distance to the transcription start site and methylation or variation in methylation? How does the number of CpG sites in the probe affect the methylation measure? Are there regional variations in methylation across the genome?

2.3 sessionInfo

R version R version 2.12.0 Under development (unstable) (2010-05-10 r51970), x86_64-apple-darwin9.8.0

locale en_US.utf-8/en_US.utf-8/C/C/en_US.utf-8/en_US.utf-8

attached base packages grid, stats, graphics, grDevices, utils, datasets, methods, base

other attached packages gplots_2.7.4, caTools_1.10, gdata_2.8.0, gtools_2.6.2, randomForest_4.5-34, GEOquery_2.13.0, RCurl_1.4-2, bitops_1.0-4.1, lattice_0.18-5, limma_3.5.3, methylumi_1.3.2, GGHumanMethCancerPanelv1.db_1.2.0, org.Hs.eg.db_2.4.1, RSQLite_0.9-0, DBI_0.2-5, AnnotationDbi_1.11.1, Biobase_2.9.0, ascii_0.6-4, proto_0.3-8

CHAPTER

THREE

THE CANCER GENOME ATLAS, GLIOBLASTOMA DATA

Author Sean Davis

Contact sdavis2@mail.nih.gov

date 2010-06-28

3.1 Background

Approximately 85-90% of all primary central nervous system tumors arise in the brain [1]. The annual incidence of all brain tumors is about 6-7 per 100,000 persons per year with a mortality of about 5 per 100,000 persons per year. Glioblastoma multiforme is the most common brain tumor in humans, accounting for about 15% of all brain tumors, and generally affects adults, though children may also develop these tumors, also. The peak incidence is around 60 years of age. The disease is, in general, devastating with a mean survival of less than one year. Surgery and radiation are the primary therapeutic modalities though chemotherapy may be used for diffuse disease such as leptomeningeal seeding or positive CSF.

From the Cancer Genome Atlas website:

> The Cancer Genome Atlas (TCGA) is a comprehensive and coordinated effort to accelerate our understanding of the molecular basis of cancer through the application of genome analysis technologies, including large-scale genome sequencing. TCGA is a joint effort of the National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI), two of the 27 Institutes and Centers of the National Institutes of Health, U.S. Department of Health and Human Services.
>
> TCGA started as a pilot project in 2006 to assess and validate the feasibility of a full-scale effort to systematically explore the entire spectrum of genomic changes involved in human cancers. With the success of the pilot project, TCGA now will expand its efforts to aggressively pursue 20 or more additional cancers to yield a comprehensive, rigorous and publicly accessible data set that will improve our ability to diagnose, treat and prevent cancer.

GBM is one of the tumors profiled in the pilot study. As such, there is quite a bit of profiling data available. I have compiled a subset of these data for experimentation.

Comment: Since this is the last lab of the course, *please experiment*, use the help() function liberally to learn more about the objects, classes, and functions being used. Also, if there are biological tangents to follow, please do so.

This vignette is designed to use the *TCGAGBM* data package. It should be installed prior to getting started.

[1] See http://cancer.gov/ for details

3.2 The Data

Based on the availability of data for the 27k Illumina methylation platform, data were downloaded from the TCGA website. The data are available in the TCGAGBM data package in the extdata directory. The clinical data will are available, also. Here is a quick overview:

```
> library(TCGAGBM)
> clinical = read.delim(system.file("extdata/TCGA_GBM_Clinical/clinical_patient_public_GBM.txt.gz",
+     package = "TCGAGBM"), header = TRUE)
> rownames(clinical) = clinical[, 1]
> summary(clinical)
   BCRPATIENTBARCODE TUMORTISSUESITE    GENDER
 TCGA-02-2466: 1      BRAIN:44        FEMALE:16
 TCGA-02-2470: 1      null : 1        MALE  :28
 TCGA-02-2483: 1                      null  : 1
 TCGA-02-2485: 1
 TCGA-02-2486: 1
 TCGA-06-2557: 1
 (Other)     :39
 PRETREATMENTHISTORY
 NO  :43
 null: 2




                       HISTOLOGICALTYPE
 null                          : 3
 Untreated primary (De Nova) GBM:42





 PRIORGLIOMA   VITALSTATUS  DAYSTOBIRTH
 NO  :44     DECEASED:24   null    : 2
 null: 1     LIVING  :19   -12060  : 1
             null    : 2   -12685  : 1
                           -13368  : 1
                           -15964  : 1
                           -16301  : 1
                           (Other):38
  DAYSTODEATH DAYSTOLASTFOLLOWUP
 null   :21   316    : 2
 133    : 1   86     : 2
 142    : 1   null   : 2
 149    : 1   -106   : 1
 15     : 1   13     : 1
 182    : 1   133    : 1
 (Other):19   (Other):36
 INFORMEDCONSENTVERIFIED
 YES:45
 AGEATINITIALPATHOLOGICDIAGNOSIS RADIATIONTHERAPY
 53     : 4                      NO  : 3
 59     : 3                      null: 3
 65     : 3                      YES :39
 52     : 2
 63     : 2
 64     : 2
 (Other):29
 CHEMOTHERAPY IMMUNOTHERAPY HORMONALTHERAPY
 NO  : 7      NO  :40       NO  :26
 null: 5      null: 4       null: 4
 YES :33      YES : 1       YES :15



 TARGETEDMOLECULARTHERAPY DAYSTOTUMORPROGRESSION
 NO  :39                  null   :28
 null: 5                  119    : 1
 YES : 1                  128    : 1
                          135    : 1
                          143    : 1
                          153    : 1
                          (Other):12
 DAYSTOTUMORRECURRENCE SITEOFTUMORFIRSTRECURRENCE
 78  : 1               null:45
 null:44




 TUMORSTAGE TUMORGRADE RESIDUALTUMOR
 null:45    null:45    null:45




 TUMORRESIDUALDISEASE
 null:45




 PRIMARYTHERAPYOUTCOMESUCCESS
 null:45
```

```
ADDITIONALRADIATIONTHERAPY
NO  :39
null: 2
YES : 4




ADDITIONALCHEMOTHERAPY ADDITIONALIMMUNOTHERAPY
NO  :33                NO  :43
null: 2                null: 2
YES :10




ADDITIONALHORMONETHERAPY ADDITIONALDRUGTHERAPY
NO  :43                  NO  :42
null: 2                  null: 3





ANATOMICORGANSUBDIVISION
Brain:39
null : 6





             INITIALPATHOLOGICDIAGNOSISMETHOD
EXCISIONAL BIOPSY            : 4
FINE NEEDLE ASPIRATION BIOPSY: 1
null                         : 1
TUMOR RESECTION              :39



PERSONNEOPLASMCANCERSTATUS    X
null      : 7              Mode:logical
TUMOR FREE: 1              NA's:45
WITH TUMOR:37
```

3.2.1 Methylation data preparation

The methylation data were not in any standard Illumina data dump format, so some custom code is necessary to convert to a MethyLumiSet object.

```
> tmp = sapply(dir(system.file("extdata/TCGA_GBM_IlluminaMethylation27k",
+     package = "TCGAGBM"), pattern = "Methylation27.7"),
+     function(x) {
+         message(x)
+         read.delim(file.path(system.file("extdata/TCGA_GBM_IlluminaMethylation27k",
+             package = "TCGAGBM"), x),
+              header = TRUE, skip = 1)
+     }, simplify = FALSE)
> matrixnames = colnames(tmp[[1]])
> matrixlist = lapply(matrixnames, function(x) {
+     message(x)
+     sapply(tmp, function(y) y[, x])
+ })
> names(matrixlist) <- matrixnames
> b = matrixlist$Methylated_Signal_Intensity..M./(matrixlist$Methylated_Signal_Intensity..M. +
+     matrixlist$Un.Methylated_Signal_Intensity..U. +
+     100)
> methTCGA = new("MethyLumiSet", betas = b)
> for (x in names(matrixlist)) {
+     message(x)
+     assayDataElement(methTCGA, x) = matrixlist[[x]]
+ }
> featureNames(methTCGA) <- assayDataElement(methTCGA,
+     "Composite.Element.REF")[, 1]
> sampleNames(methTCGA) = substr(sub("jhu-usc.edu_GBM.HumanMethylation27.7.lvl-1.",
+     "", sampleNames(methTCGA)), 1, 12)
> pData(methTCGA) = clinical[match(clinical[,
+     1], sampleNames(methTCGA)), ]
> annotation(methTCGA) = "IlluminaHumanMethylation27k"
> experimentData(methTCGA) = new("MIAME",
+     name = "TCGA GBM Methylation 27k data",
+     lab = "TCGA, JHU methylation", url = "http://cancergenome.nih.gov/")
```

Now, to look at what we have just loaded:

```
> methTCGA
Object Information:
MethyLumiSet (storageMode: lockedEnvironment)
assayData: 27578 features, 45 samples
  element names: betas, Composite.Element.REF, Detection_P_Value, M_Number_Beads, M_STDERR, Methylate
protocolData: none
phenoData
  sampleNames: TCGA-02-2466, TCGA-02-2470, ..., TC
  GA-32-1986  (45 total)
  varLabels and varMetadata description:
    BCRPATIENTBARCODE: NA
    TUMORTISSUESITE: NA
    ...: ...
    X: NA
    (34 total)
featureData: none
experimentData: use 'experimentData(object)'
Annotation: IlluminaHumanMethylation27k
Major Operation History:
[1] submitted finished  command
<0 rows> (or 0-length row.names)
> experimentData(methTCGA)
Experiment data
  Experimenter name: TCGA GBM Methylation 27k data
  Laboratory: TCGA, JHU methylation
  Contact information:
  Title:
  URL: http://cancergenome.nih.gov/
```

```
PMIDs:
No abstract available.
```

3.2.2 CGH data preparation

Data from the same patients as above were downloaded from the TCGA website. Briefly, the samples were run on 244k Agilent long oligo arrays with Promega commercial DNA run as a reference. We can use the limma package to load and manipulate the data.

```
> library(limma)
> tmp2 = read.maimages(files = dir(system.file("extdata/TCGA_GBM_244kcgh",
+     package = "TCGAGBM"), pattern = "MSK"),
+     path = system.file("extdata/TCGA_GBM_244kcgh",
+         package = "TCGAGBM"), source = "agilent")
```

A little convenience function, splitAgilentChromLocs is useful to get the chromosome locations from the feature extraction data.

```
> splitAgilentChromLocs <- function(systematicName) {
+     tmp <- gsub("[:-]", ":", as.character(systematicName))
+     tmp2 <- data.frame(do.call(rbind,
+         strsplit(as.character(tmp), ":")))
+     colnames(tmp2) <- c("chrom", "start",
+         "end")
+     tmp2[, 2] = as.integer(as.character(tmp2[,
+         2]))
+     tmp2[, 3] = as.integer(as.character(tmp2[,
+         3]))
+     return(tmp2)
+ }
> locs = splitAgilentChromLocs(as.character(tmp2$genes$SystematicName))
> tmp2$genes = data.frame(tmp2$genes, locs)
```

Remove the control probes and “normalize”–that is, convert to MA from RG.

```
> tmp2 = tmp2[tmp2$genes$ControlType ==
+     0, ]
> cghTCGAMA = normalizeWithinArrays(tmp2,
+     method = "none", bc.method = "none")
```

Now, map the sample names back to the arrays.

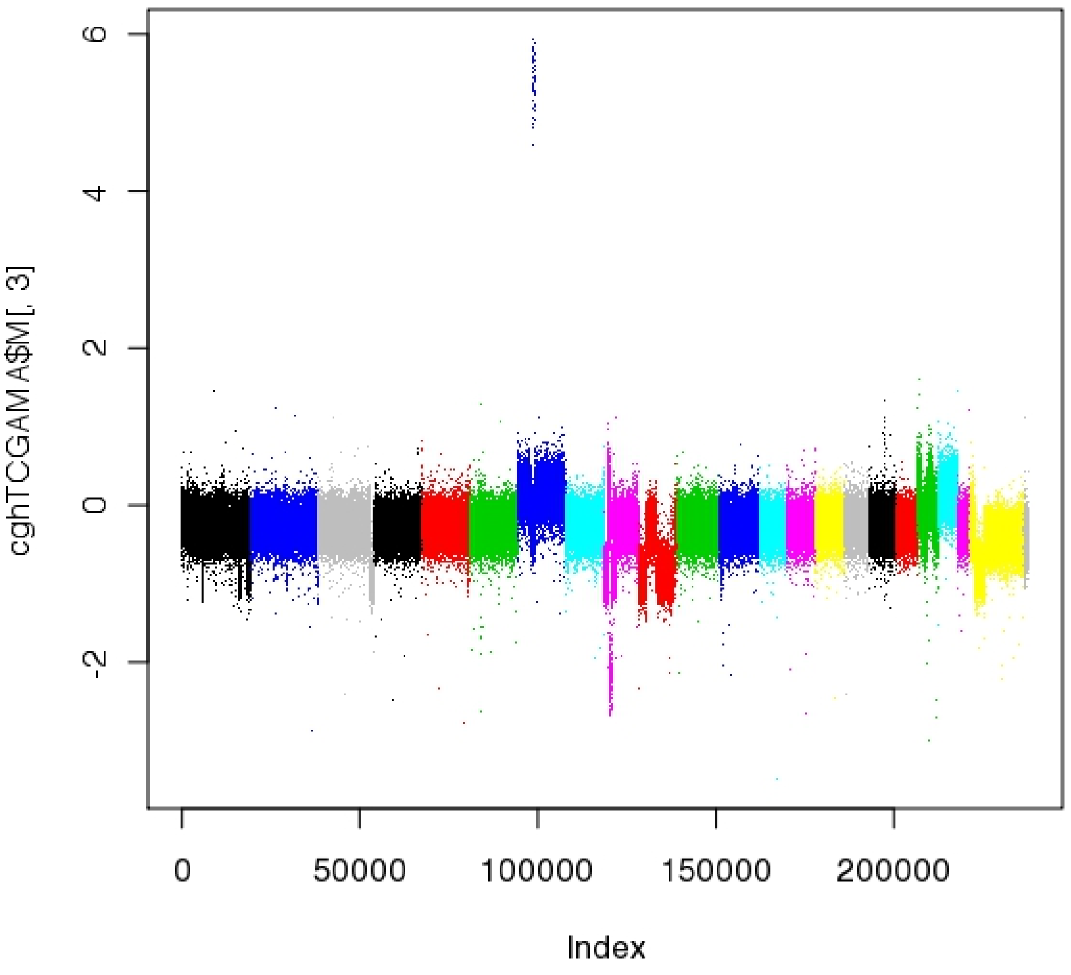
```
> cghsdrf = read.delim(system.file("extdata/TCGA_GBM_244kcgh/mskcc.org_GBM.HG-CGH-244A.1.sdrf.txt",
+     package = "TCGAGBM"))
> cghsdrf = cghsdrf[cghsdrf$Provider ==
+     "BCR", ]
> cghTCGAMA$targets$sample = substr(cghsdrf[match(rownames(cghTCGAMA$targets),
+     cghsdrf$Array.Data.File), 1], 1, 12)
> cghTCGAMA$targets = data.frame(cghTCGAMA$targets,
+     clinical[match(cghTCGAMA$targets$sample,
+         clinical[, 1]), ])
```

And finally, just to make things easier in the future, order the probes in chromosome order and remove probes mapping to odd places like chr7_random, etc.

```
> numericchrom = sub("chr", "", cghTCGAMA$genes$chrom)
> numericchrom[numericchrom == "X"] = 23
> numericchrom[numericchrom == "Y"] = 24
> numericchrom = as.integer(numericchrom)
> cghTCGAMA = cghTCGAMA[!is.na(numericchrom),
+     ]
> numericchrom = numericchrom[!is.na(numericchrom)]
> cghTCGAMA = cghTCGAMA[order(numericchrom,
+     cghTCGAMA$genes$start), ]
> cghTCGAMA$genes$chrom = factor(as.character(cghTCGAMA$genes$chrom))
> cghTCGAMA$genes$Chr = numericchrom
> cghTCGAMA$genes$Position = as.numeric(cghTCAGAMA$genes$start)
```

Just to get a rough idea of how things look, let’s make a plot of one of the samples:

```
> plot(cghTCGAMA$M[, 3], pch = ".", col = cghTCGAMA$genes$chrom)
```



3.2.3 Expression data preparation

Again, the data were downloaded from the TCGA website. A rather unusual array platform was used for the gene expression analysis for which there was actually no array description file available. The array is a custom Agilent 244k array with multiple probes per gene. However, the TCGA project does make available processed data for these samples. It appears that the processing was pretty standard with the multiple probes per gene averaged and then loess normalized. In any case, for our purposes, the data are probably good enough for comparison to other data types.

```
> dat1 = read.delim(system.file("extdata/TCGA_GBM_geneexp/unc.edu__AgilentG4502A_07_2__gene_expressio
+     package = "TCGAGBM"), header = TRUE)
> datmat = matrix(as.numeric(as.character(dat1$value)),
+     nrow = nrow(dat1)/length(unique(dat1$barcode)))
> colnames(datmat) = substr(unique(dat1$barcode),
+     1, 12)
> rownames(datmat) = dat1$gene.symbol[1:(nrow(dat1)/length(unique(dat1$barcode)))]
> expTCGA = new("ExpressionSet", exprs = datmat)
> experimentData(expTCGA) = new("MIAME",
+     name = "TCGA GBM level 3 expression data",
+     lab = "TCGA, UNC gene expression",
+     url = "http://cancergenome.nih.gov/")
> pData(expTCGA) = clinical[match(sampleNames(expTCGA),
+     clinical[, 1]), ]
```

3.3 Analyses

Now that the data are loaded and organized, it is time to begin some analysis. Note that I supply some suggested workflows, but there are plenty of tangents that might be worth following.

3.3.1 Expression and Methylation Correlation

It is interesting, of course, to examine directly the effects of DNA methylation on gene expression. Armed with our DNA methylation data and our gene expression data, we can directly calculate correlations between the two data types. In order to perform the calculations, we will want to generate two matrices, one for gene expression and one for DNA methylation. Each matrix will need to have the same ordering of samples so that we can calculate on them directly. Also, we will want to map the DNA methylation probes to their matching gene expression probes. Recall that the gene expression data are in the form on an ExpressionSet and that the featureNames of the ExpressionSet are the actual HGNC gene names.

```
> data(expTCGA)
> featureNames(expTCGA)[10:20]
 [1] "A4GNT"    "AAAS"     "AACS"     "AADAC"
 [5] "AADACL1" "AADACL2" "AADACL3" "AADACL4"
 [9] "AADAT"    "AAK1"     "AAMP"
```

For the methylation data, on the other hand, the annotation for the probes is not directly available from the MethyLumiSet object, methTCGA. Instead, the featureNames are Illumina probe names.

```
> data(methTCGA)
> featureNames(methTCGA)[10:20]
 [1] "cg00010193" "cg00011459" "cg00012199"
 [4] "cg00012386" "cg00012792" "cg00013618"
 [7] "cg00014085" "cg00014837" "cg00015770"
[10] "cg00016968" "cg00019495"
```

The appropriate data package is the IlluminaHumanMethylation27k.db package. It may be necessary to install it using biocLite if it is not already installed. Attempting to load the package will show if it is installed.

```
> require(IlluminaHumanMethylation27k.db)
```

As with other annotation packages, we can quickly look to see what is inside:

```
> IlluminaHumanMethylation27k()
Quality control information for IlluminaHumanMethylation27k:


This package has the following mappings:

IlluminaHumanMethylation27kACCNUM has 27551 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kALIAS2PROBE has 57212 mapped keys (of 110391 keys)
IlluminaHumanMethylation27kCHR has 25976 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kCHRLENGTHS has 93 mapped keys (of 93 keys)
IlluminaHumanMethylation27kCHRLOC has 25948 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kCHRLOCEND has 25948 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kENSEMBL has 25860 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kENSEMBL2PROBE has 14496 mapped keys (of 19971 keys)
IlluminaHumanMethylation27kENTREZID has 25976 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kENZYME has 3467 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kENZYME2PROBE has 855 mapped keys (of 912 keys)
IlluminaHumanMethylation27kGENENAME has 25976 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kGO has 24428 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kGO2ALLPROBES has 12023 mapped keys (of 12327 keys)
IlluminaHumanMethylation27kGO2PROBE has 8841 mapped keys (of 9168 keys)
IlluminaHumanMethylation27kMAP has 25911 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kOMIM has 19955 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kPATH has 8131 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kPATH2PROBE has 200 mapped keys (of 200 keys)
IlluminaHumanMethylation27kPFAM has 25957 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kPMID has 25964 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kPMID2PROBE has 250823 mapped keys (of 265573 keys)
IlluminaHumanMethylation27kPROSITE has 25957 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kREFSEQ has 25976 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kSYMBOL has 25976 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kUNIGENE has 25959 mapped keys (of 27578 keys)
IlluminaHumanMethylation27kUNIPROT has 25906 mapped keys (of 27578 keys)


Additional Information about this package:

DB schema: HUMANCHIP_DB
DB schema version: 2.1
Organism: Homo sapiens
Date for NCBI data: 2010-Mar1
Date for GO data: 20100320
Date for KEGG data: 2010-Feb28
Date for Golden Path data: 2009-Jul5
Date for IPI data: 2010-Feb10
Date for Ensembl data: 2010-Mar3
```

Since we want to map methylation and gene expression to each other, we can use the annotation package to get gene names for the methylation data.

```
> methgenenames = unlist(mget(featureNames(methTCGA),
+     IlluminaHumanMethylation27kSYMBOL,
+     ifnotfound = NA))
> sum(is.na(methgenenames))
[1] 1611
```

There are 1611 probes on the methylation platform that, for whatever reason, do not have associated gene symbols. We will simply exclude those from downstream analysis. The mapping between platforms is now pretty straightforward.

```
> tmp = match(methgenenames, featureNames(expTCGA))
> methdat = betas(methTCGA)[!is.na(tmp),
+     order(sampleNames(methTCGA))]
> expdat = exprs(expTCGA)[tmp[!is.na(tmp)],
+     order(sampleNames(expTCGA))]
```

Here, we are relying on the fact that the sample names are the same between the two data sets so that ordering by sampleNames will result in matching orders. We can quickly double-check that that is the case.

```
> match(colnames(expdat), colnames(methdat))
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
[16] 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
[31] 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
> dim(expdat)
[1] 23904    45
> dim(methdat)
[1] 23904    45
```

The sample names appear to match and the matrices are the same size. Now, we are ready to calculate some correlations. The statement below simply does a row wise Pearson correlation calculation. The result will be one correlation value per row of data, each of which corresponds to a methylation probe and its associated expression probe. Note that each expression probe may map to several methylation probes.

```
> x = sapply(1:nrow(methdat), function(i) cor(methdat[i,
+     ], expdat[i, ]))
```

So, what would we expect this distribution to look like?

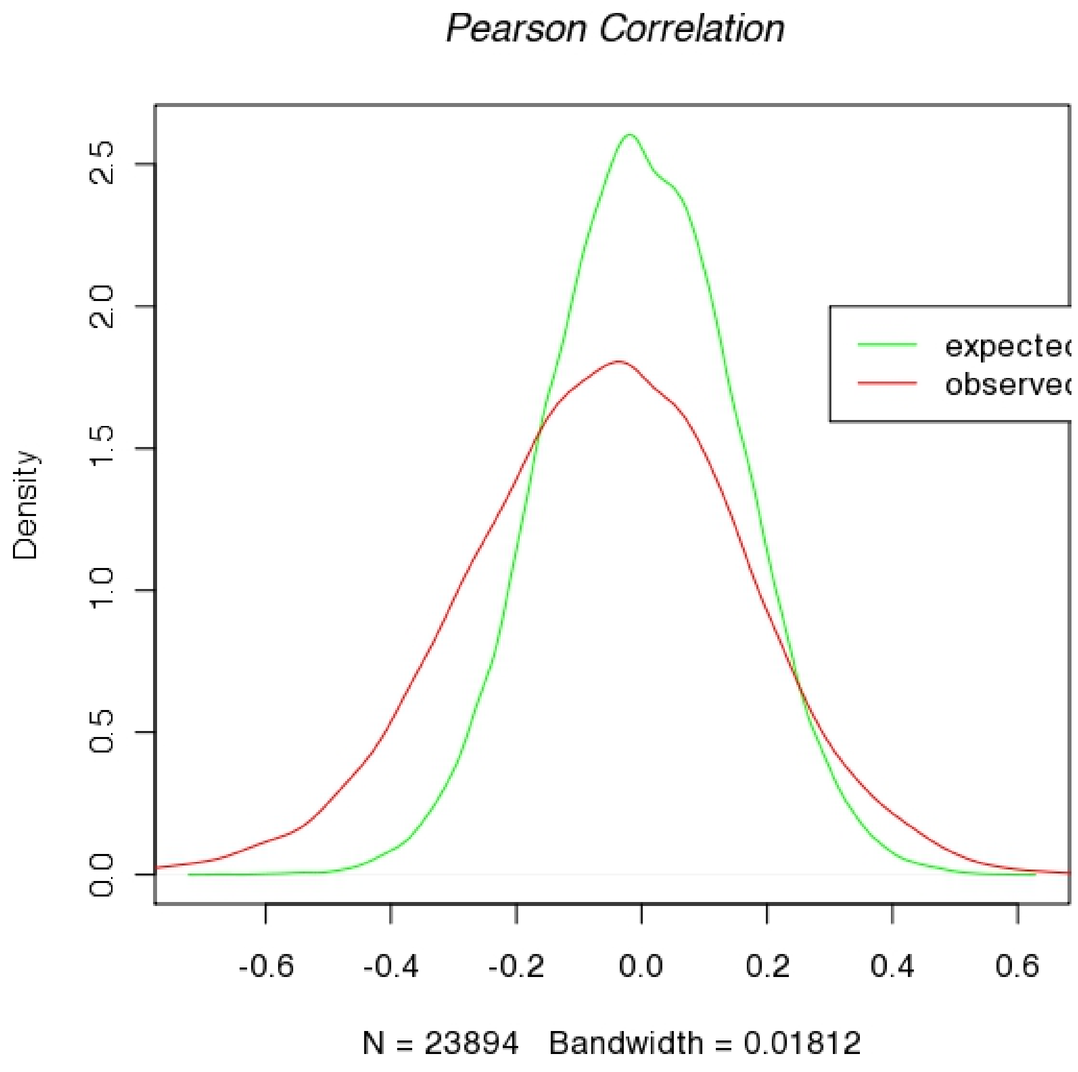
```
> summary(x)
    Min.  1st Qu.   Median     Mean  3rd Qu.
-0.92580 -0.20590 -0.05306 -0.05881  0.09256
    Max.     NA's
 0.83900 10.00000
```

The mean is less than 0 and there is a definite skew to the left. How big is the effect? In order to determine what the distribution might look like under the null hypothesis, we can simply permute one of the sample sets relative to the other, effectively breaking any correlation that might exist.

```
> y = sapply(1:nrow(methdat), function(i) cor(methdat[i,
+     ], expdat[i, sample(1:45, 45)]))
> summary(y)
      Min.    1st Qu.     Median       Mean
-0.6681000 -0.1037000 -0.0019640 -0.0005753
   3rd Qu.       Max.       NA's
 0.1030000  0.5739000 10.0000000
```

Though this is only one replicate of randomization, it is obvious that this distribution looks more like what we would expect if there were no effect of DNA methylation on gene expression. A quick plot is probably useful to examine the differences more globally.

```
> plot(density(y[!is.na(y)]), col = "green",
+     main = "Pearson Correlation")
> lines(density(x[!is.na(x)]), col = "red")
> legend(0.3, 2, legend = c("expected",
+     "observed"), col = c("green", "red"),
+     lty = 1)
```



Note the shoulder to the left in the ‘observed’ data, indicating an effect of DNA methylation on gene expression. One could also do some hypothesis testing to determine if the effect is statistically significant.

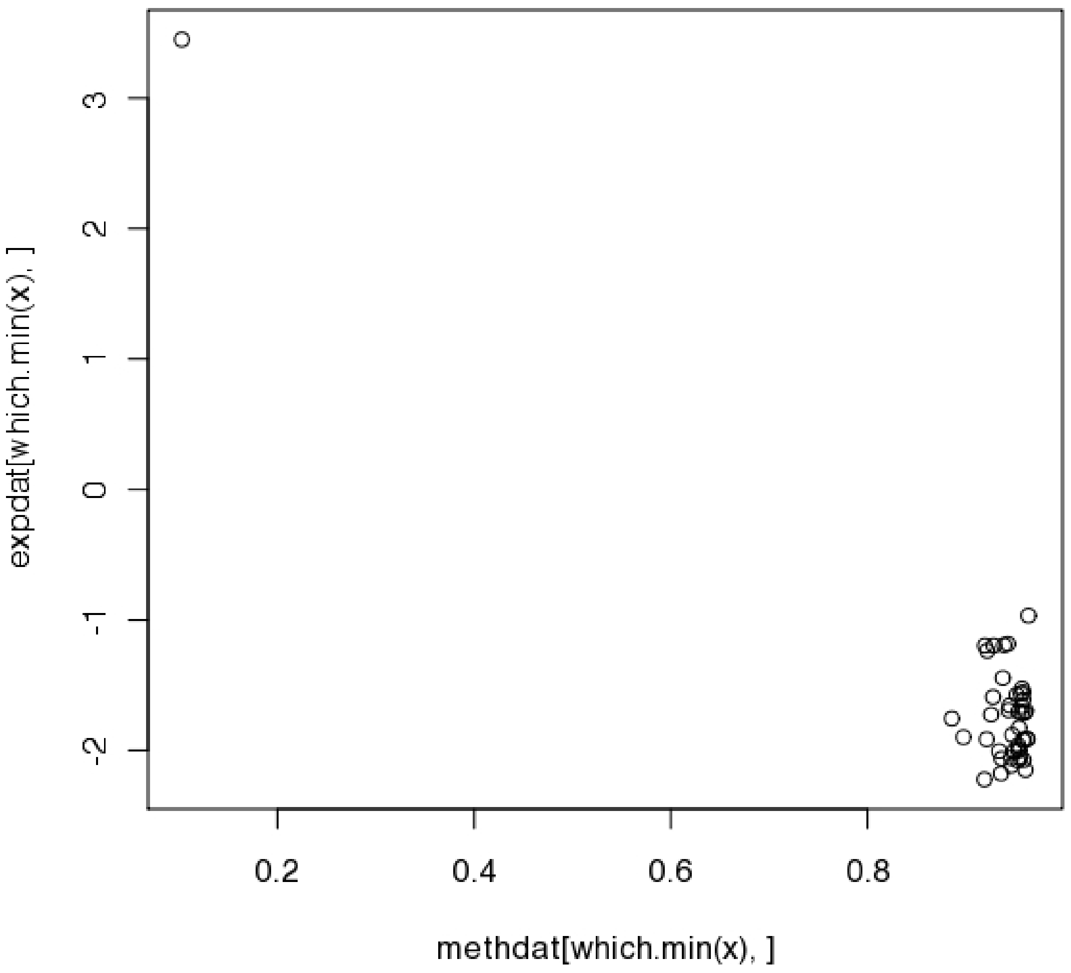
There appears to be a global effect of DNA methylation on gene expression. Which genes show the largest effect?

```
> mincor = rownames(methdat)[which.min(x)]
> mget(mincor, IlluminaHumanMethylation27kSYMBOL)
```

```
$cg15933546
[1] "DNAH8"
```

And what does a plot of gene expression and DNA methylation look like for that probe?

```
> plot(methdat[which.min(x), ], expdat[which.min(x),
+     ])
```



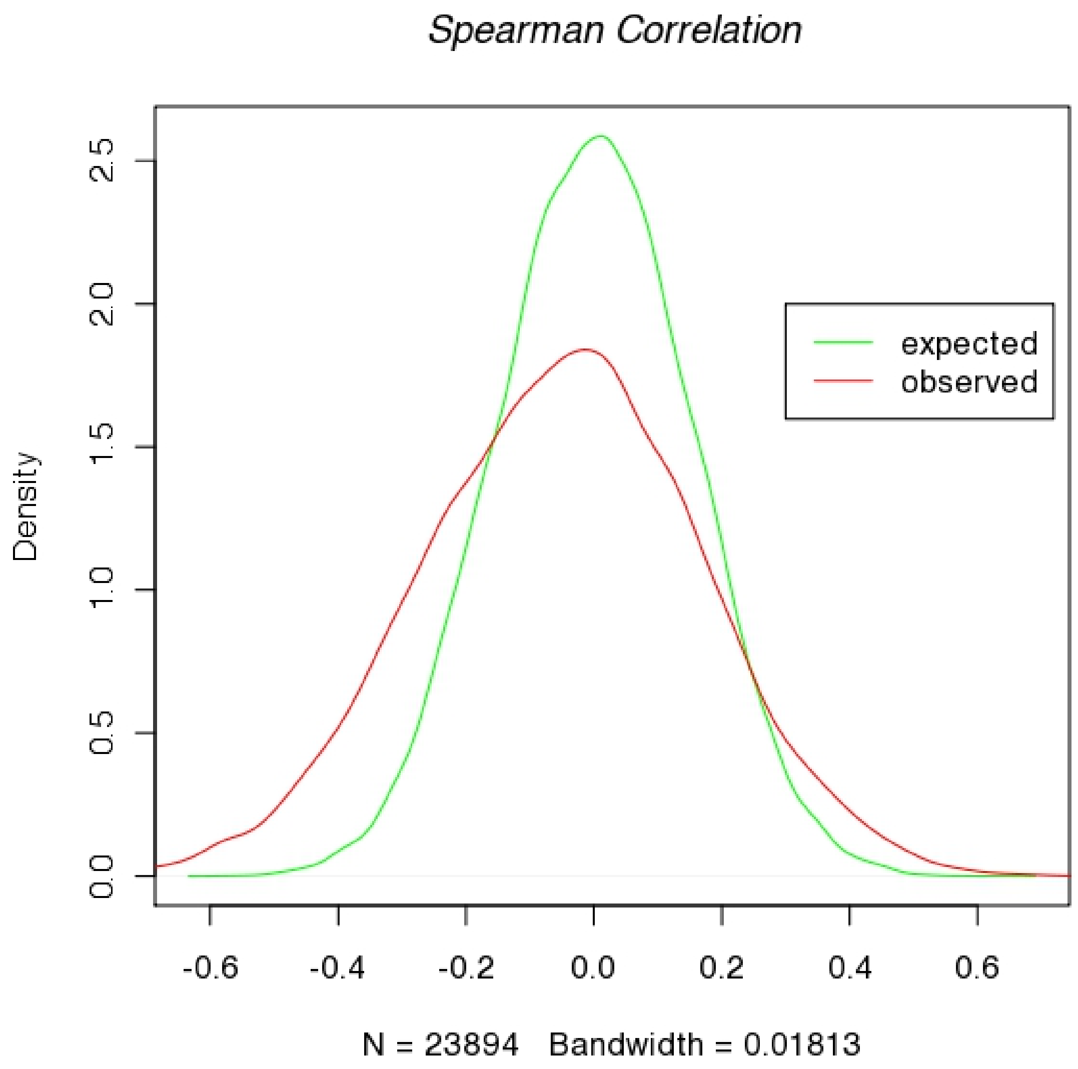
So, there appears to be a bit of a problem with our correlation measure. The Pearson correlation measure will find such outliers quite nicely. Instead, perhaps we should use a rank-based correlation metric.

```
> x = sapply(1:nrow(methdat), function(i) cor(methdat[i,
+     ], expdat[i, ], method = "spearman"))
> summary(x)
    Min.  1st Qu.   Median     Mean  3rd Qu.
-0.82600 -0.20010 -0.04559 -0.05110  0.09796
    Max.     NA's
```

```
 0.72650 10.00000
> y = sapply(1:nrow(methdat), function(i) cor(methdat[i,
+     ], expdat[i, sample(1:45, 45)], method = "spearman"))
> summary(y)
      Min.    1st Qu.     Median       Mean 
-0.5791000 -0.1024000  0.0010210  0.0001465 
   3rd Qu.       Max.       NA's 
 0.1028000  0.6362000 10.0000000 
```

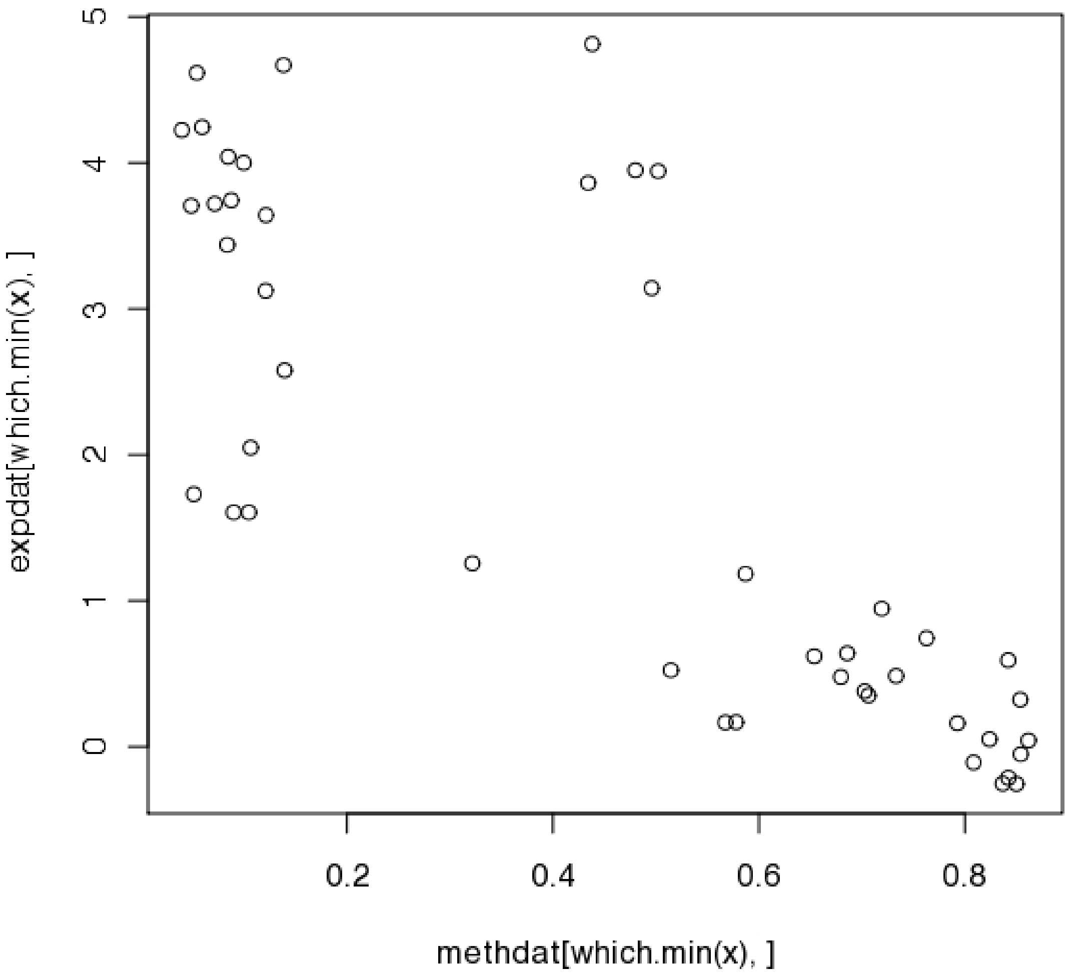
The same general trend seems to be present using the Spearman correlation coefficient as with the Pearson correlation coefficient.

```
> plot(density(y[!is.na(y)]), col = "green",
+     main = "Spearman Correlation")
> lines(density(x[!is.na(x)]), col = "red")
> legend(0.3, 2, legend = c("expected",
+     "observed"), col = c("green", "red"),
+     lty = 1)
```



And the plot looks hearly identical. Did we do any better with finding biologically meaningful top methylation candidates?

```
> plot(methdat[which.min(x), ], expdat[which.min(x),
+     ])
```



That looks better. The shape of the plot is interesting. Except for five data points, there appears to be a threshold effect whereby genes are more highly expressed only when methylation is very low.

- Try writing a short function that will take as input an integer that represents the rank in correlation (lowest being the first), the correlation vector, and the two data matrices and produce a plot of the data for that probe.
- Extend the function to include the gene name in the plot.
- Instead of returning the correlation coefficient like above, change the code so that a p.value is returned. *Hint*: try experimenting with the cor.test function.

Questions for thought:

- How would you go about determining if there is a regional bias in gene expression and DNA methylation?

3.3.2 CGH Data Analysis

CGH data are unlike many other types of high-throughput data such as gene expressin or DNA methylation. The fact that copy number in the genome is *piecewise continuous* is used by copy number segmentation methods to take measurements that are noisy at the individual probe level and *smooth* them in a non-continuous way; that is, these methods try to respect natural changepoints in the data. We will be using the snapCGH package to facilitate some of these analyses. Make sure that it is installed.

```
> require(snapCGH, quiet = TRUE)
```

If this doesn't work, go ahead and install using biocLite().

Quality control issues

I had already preprocessed the data above into a useful form. The data set is in the form of a limma MAList. The snapCGH package takes that as an input.

```
> library(limma)
> require(snapCGH)
> data(cghTCGAMA)
> class(cghTCGAMA)
[1] "MAList"
attr(,"package")
[1] "limma"
```

As you can see, the data just loaded are in the form of a limma MAList. The M values in this MAList are *not* gene expression measures, but log2 ratios of tumor DNA to a reference normal DNA. For normal individuals, the ratio of sample to reference signal will be close to 1 for a log2 ratio of 0. Gain of a copy of DNA at a locus will result in a value of log2(3/2) while loss of a copy will show log2(1/2). The sex chromosomes will behave slightly differently; that behavior will depend on the genders of both the reference and the sample.

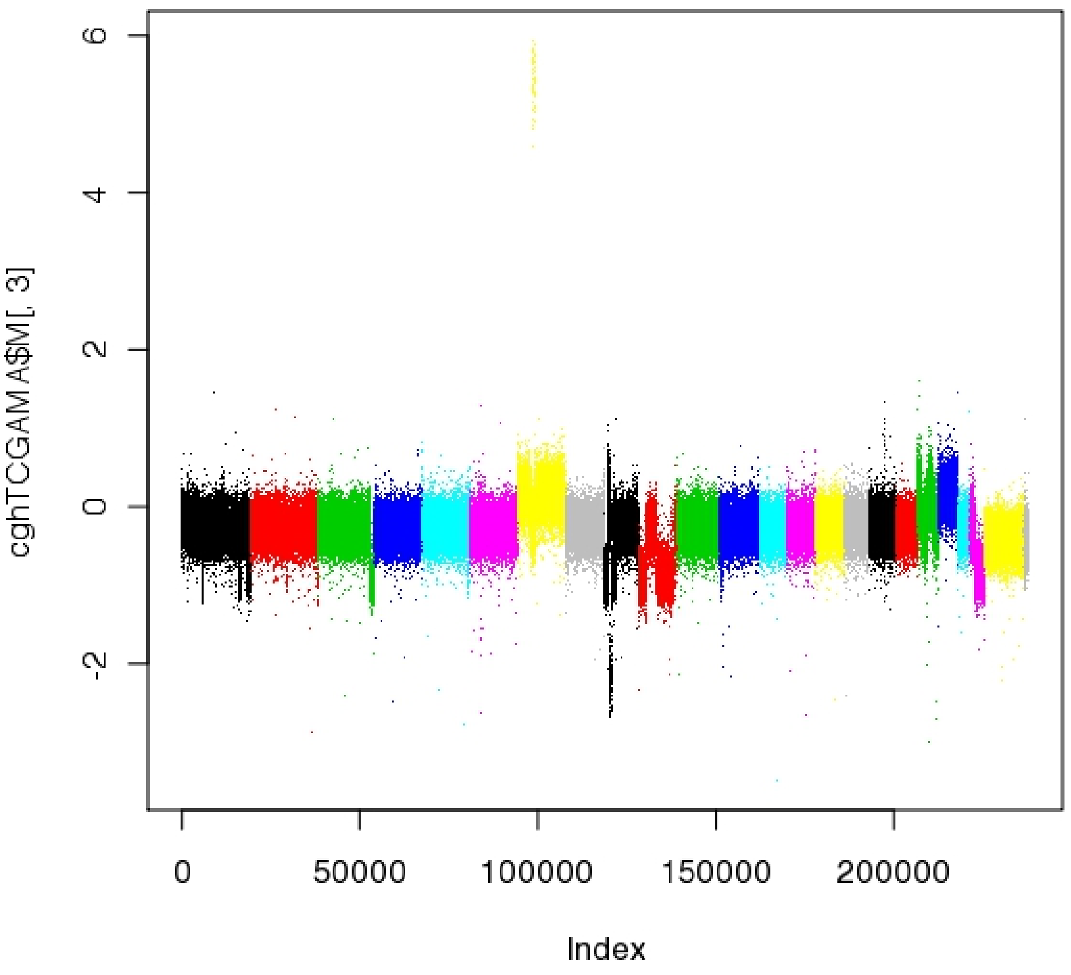
The annotation for these data are in the $genes list element.

```
> head(cghTCGAMA$genes)
       Row Col ProbeUID ControlType
8437    19 233     8589           0
25100   56  31    24444           0
61506  135 419    61188           0
48639  107 318    48398           0
69696  153 403    69245           0
107488 236 354   105927           0
            ProbeName                  GeneName
8437     A_14_P112718                  AK026901
25100  A_16_P15000916                  AK026901
61506  A_16_P15001074                  AK125248
48639  A_16_P00000012 chr1:000736483-000736542
69696  A_16_P00000014 chr1:000742533-000742586
107488 A_16_P00000017 chr1:000746956-000747005
                 SystematicName
8437   chr1:000554268-000554327
25100  chr1:000554287-000554346
61506  chr1:000639581-000639640
48639  chr1:000736483-000736542
69696  chr1:000742533-000742586
107488 chr1:000746956-000747005
                                                  Description
8437      Homo sapiens cDNA: FLJ23248 fis, clone COL03555.
25100     Homo sapiens cDNA: FLJ23248 fis, clone COL03555.
61506  Homo sapiens cDNA FLJ43258 fis, clone HHDPC1000001.
48639                                                 Unknown
```

```
69696                                                         Unknown
107488                                                        Unknown
       chrom  start    end Chr Position
8437    chr1 554268 554327   1   554268
25100   chr1 554287 554346   1   554287
61506   chr1 639581 639640   1   639581
48639   chr1 736483 736542   1   736483
69696   chr1 742533 742586   1   742533
107488  chr1 746956 747005   1   746956
```

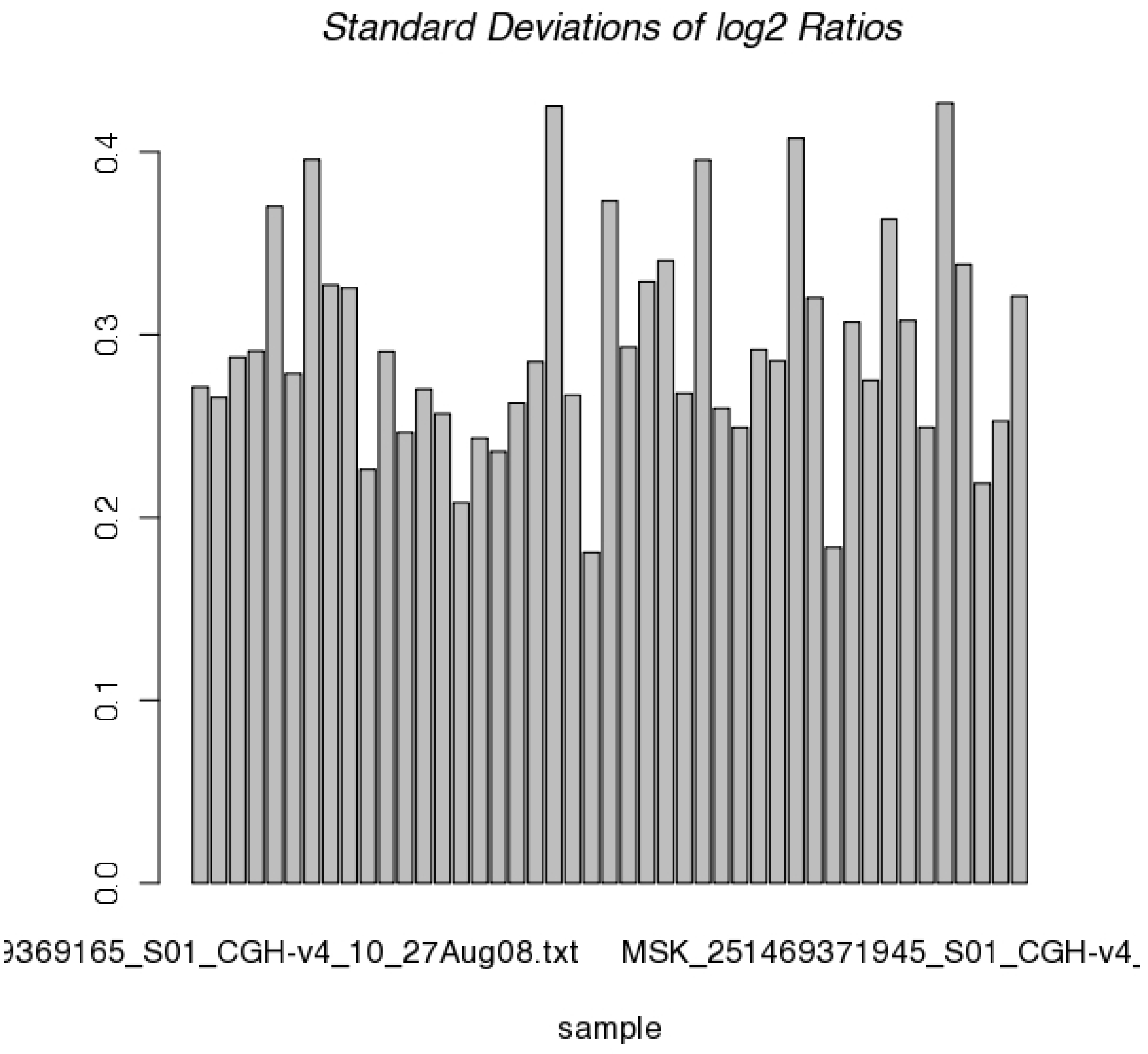
The data should be sorted by chromosome and position already, so we can plot a whole genome of log2 ratios easily with a single command. I add color to distinguish the chromosomes.

```
> plot(cghTCGAMA$M[, 3], col = cghTCGAMA$genes$Chr,
+     pch = ".")
```



You can see that most of the values are, indeed, near 0 with some spread. This spread represents the "noise" in the data and should probably be quantified. Normally, the spread of such values can be estimated by a measure such as the standard deviation. This will work just fine if the entire genome has a common mean (no copy number variation). How do the standard deviations look across our samples?

```
> log2sds = apply(cghTCGAMA$M, 2, sd)
> barplot(log2sds, main = "Standard Deviations of log2 Ratios",
+     xlab = "sample")
```

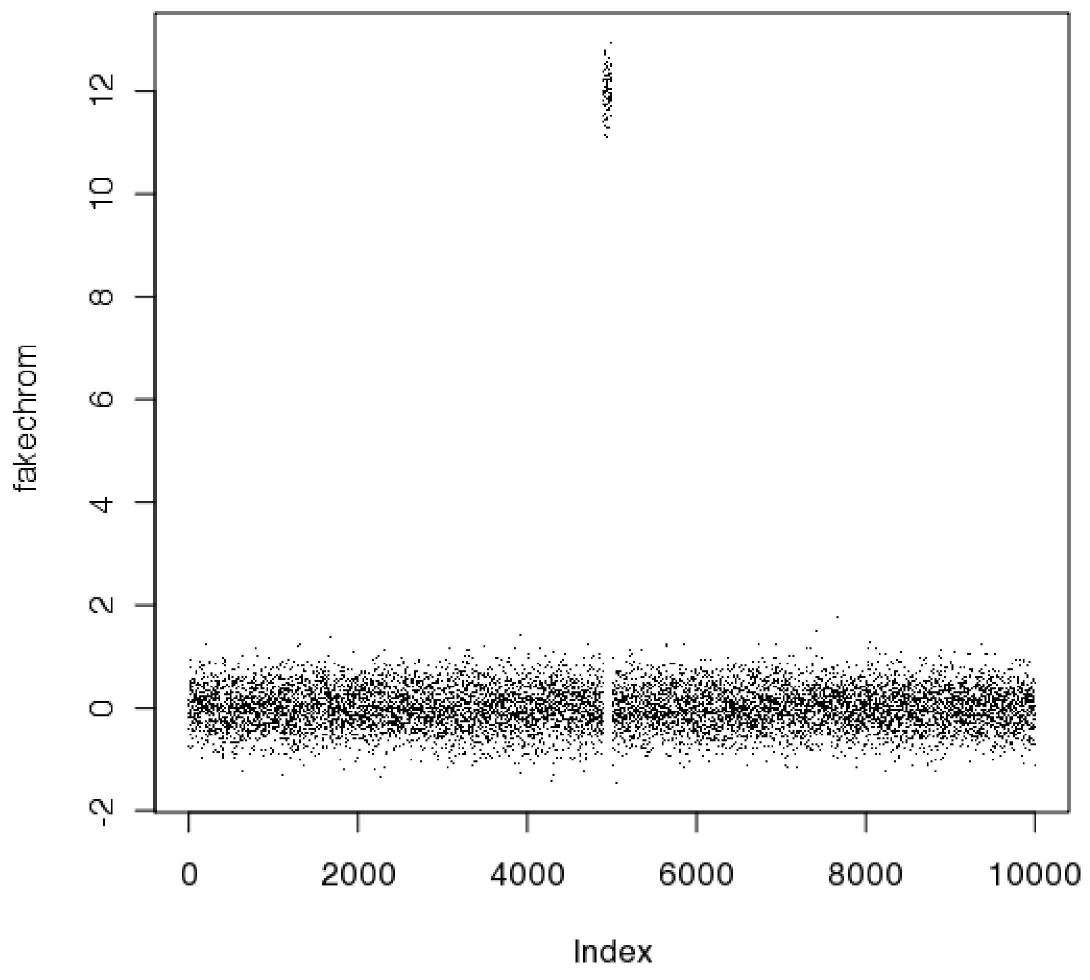


When advising experimentalists of the quality of their CGH arrays, it is important to be able to provide a robust estimate of the noise in the data. To see if the standard deviation is a very robust estimator of the noise. Let's consider a hypothetical chromosome with 10000 probes on it and no copy number variation. We can simulate such a thing quickly in R using rnorm and let's use a standard deviation of 0.4, similar to that in our data.

```
> fakechrom = rnorm(10000, sd = 0.4)
> sd(fakechrom)
[1] 0.4007763
```

Not very interesting, but let's put a copy number change in the middle of the chromosome that is similar in scale to that shown in the sample you plotted above:

```
> fakechrom[4900:5000] = fakechrom[4900:5000] +
+     4
> plot(fakechrom, pch = ".")
> sd(fakechrom)
[1] 0.5684836
```



Note that the standard deviation is now quite a bit larger than the 0.4 that we had anticipated. In other words, the standard deviation is sensitive to outliers in the data and with cancer samples, there can be quite a few outliers. With genomic data of any kind that might be expected to behave in a piecewise constant fashion, one can use another measure of noise, the derivative log ratio spread (DLRS). Defined here:

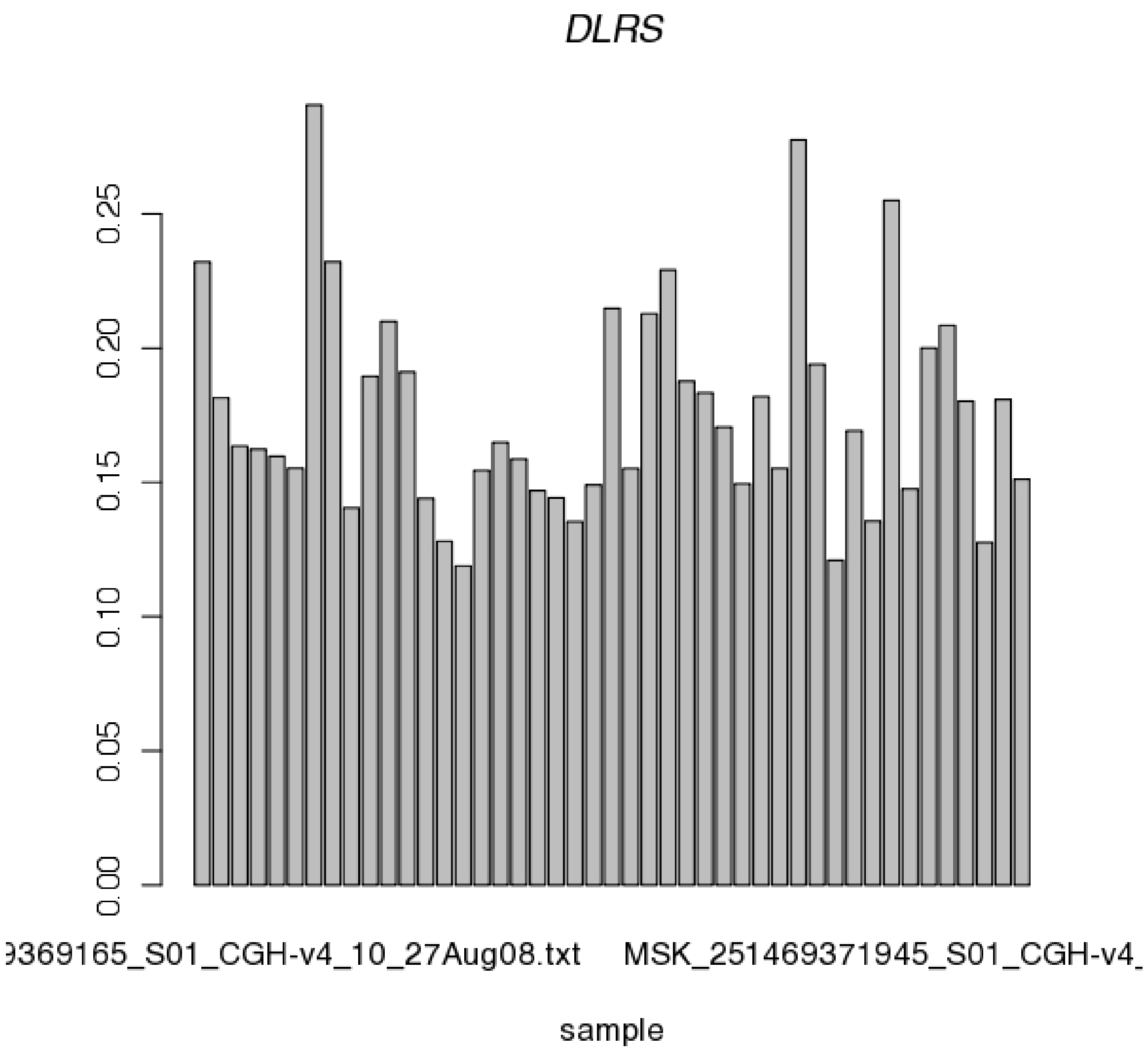
```
> dlrs <- function(x) {
+     nx <- length(x)
+     if (nx < 3) {
+         stop("Vector length>2 needed for computation")
+     }
+     tmp <- embed(x, 2)
+     diffs <- tmp[, 2] - tmp[, 1]
+     dlrs <- IQR(diffs, na.rm = TRUE)/(sqrt(2) *
+         1.34)
+     return(dlrs)
+ }
```

What is the dlrs of our fake chromosome, even with the noise added?

```
> dlrs(fakechrom)
[1] 0.4012613
```

And of our samples?

```
> log2dlrs = apply(cghTCGAMA$M, 2, dlrs)
> barplot(log2dlrs, main = "DLRS", xlab = "sample")
```



The DLRS is a useful measure of the actual noise in the data and is largely unaffected by “signal”. As such, it is a useful reporting tool for the technical quality of an array. All of the arrays in our sample set have DLRS values < 0.3, a maximum value recommended by Agilent.

We have applied the dlrs here to two-color Agilent CGH data, but such a measure could also be applied to log2 ratios from SNP arrays or even to copy number estimates generated by second-generation sequencing.

Data segmentation

A key component of making the best use of CGH data is called “segmentation’‘. There is extensive literature and literally dozens of methods for performing segmentation on CGH data. The snapCGH package provides convenient wrappers for many of these methods as implemented in R packages. One that is particularly popular, easy to use, and fairly robust with not much need for parameter tuning on this data set is Circular Binary Segmentation (CBS) and is implemented in the DNAcopy package. The snapCGH package provides a wrapper called, conveniently enough, “runDNAcopy”.

```
> require(snapCGH)
> require(DNAcopy)
> citation("DNAcopy")
To cite package 'DNAcopy' in publications
use:

  Venkatraman E. Seshan and Adam Olshen ().
  DNAcopy: DNA copy number data analysis. R
  package version 1.23.3.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {DNAcopy: DNA copy number data analysis},
    author = {Venkatraman E. Seshan and Adam Olshen},
    year = {},
    note = {R package version 1.23.3},
  }

ATTENTION: This citation information has
been auto-generated from the package
DESCRIPTION file and may need manual
editing, see 'help("citation")' .
```

The citation function can be used with any R package to get a suggested reference or references. In some cases, the authors will list publications here, also.

The snapCGH and DNAcopy packages are loaded, so we can proceed to the segmentation process. This process can be rather time-consuming, so I suggest using only a subset of about 3 samples.

```
> cghTCGAMA$design = rep(1, ncol(cghTCGAMA))
> cghTCGAMA123 = cghTCGAMA[, 1:3]
```

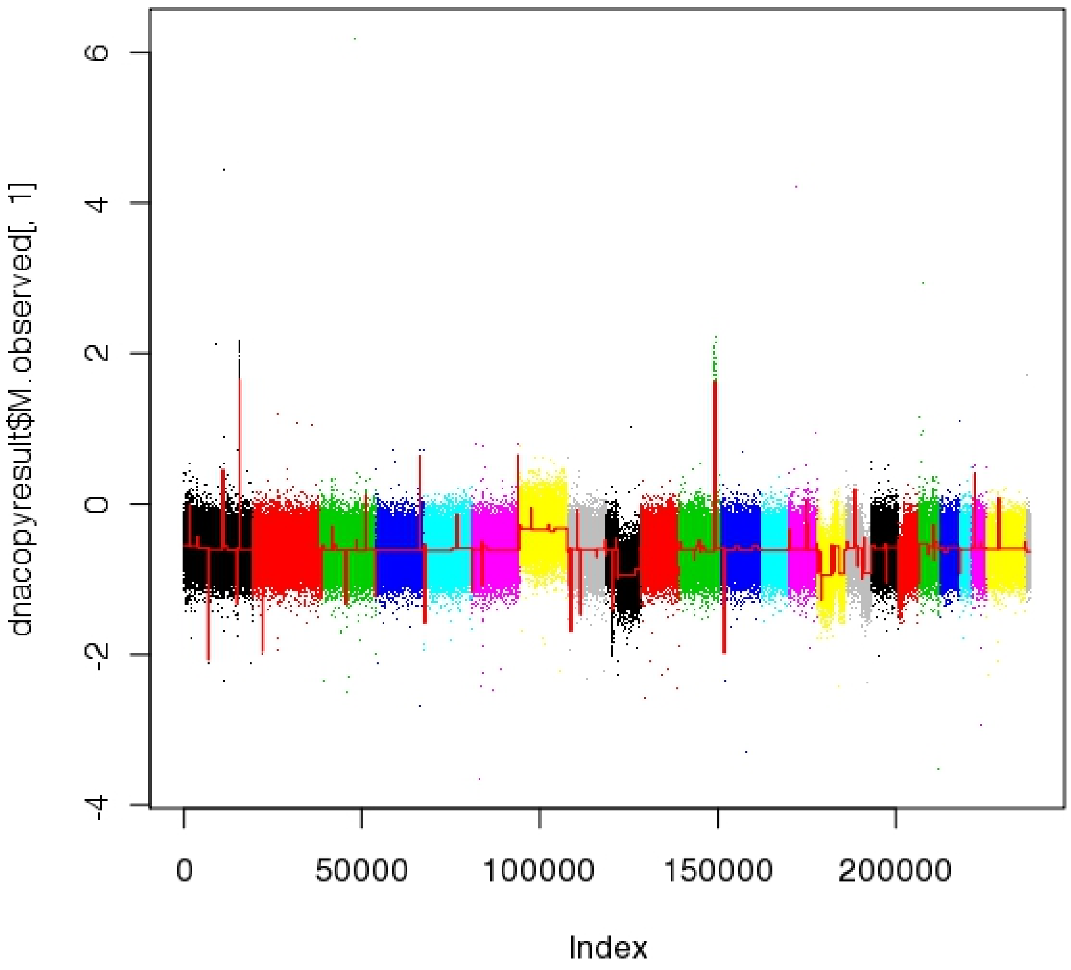
```
> dnacopyresult = runDNAcopy(cghTCGAMA123)
Analyzing: Sample.1
Analyzing: Sample.2
Analyzing: Sample.3
> dim(dnacopyresult)
[1] 237834      3
> class(dnacopyresult)
```

```
[1] "SegList"
attr(,"package")
[1] "snapCGH"
```

Warning messages are OK here. Error messages are not.

The interesting part of the dnacopyresult object is the M.predicted list element. Perhaps a plot is the most interesting way to look at the data. First, plot the raw data which is, for convenience, included in the dnacopyresult object in the M.observed list element. We add to this plot the estimated log2 ratio estimates from the DNAcopy segmentation.

```
> plot(dnacopyresult$M.observed[, 1], col = dnacopyresult$genes$Chr,
+     pch = ".")
> lines(dnacopyresult$M.predicted[, 1],
+     col = "red")
```



At this point, you have successfully segmented (smoothed) your data. The snapCGH package has other options for segmentation besides runDNAcopy. Feel free to experiment by using the help function to get other options and learn how to use them.

Centering the CGH data

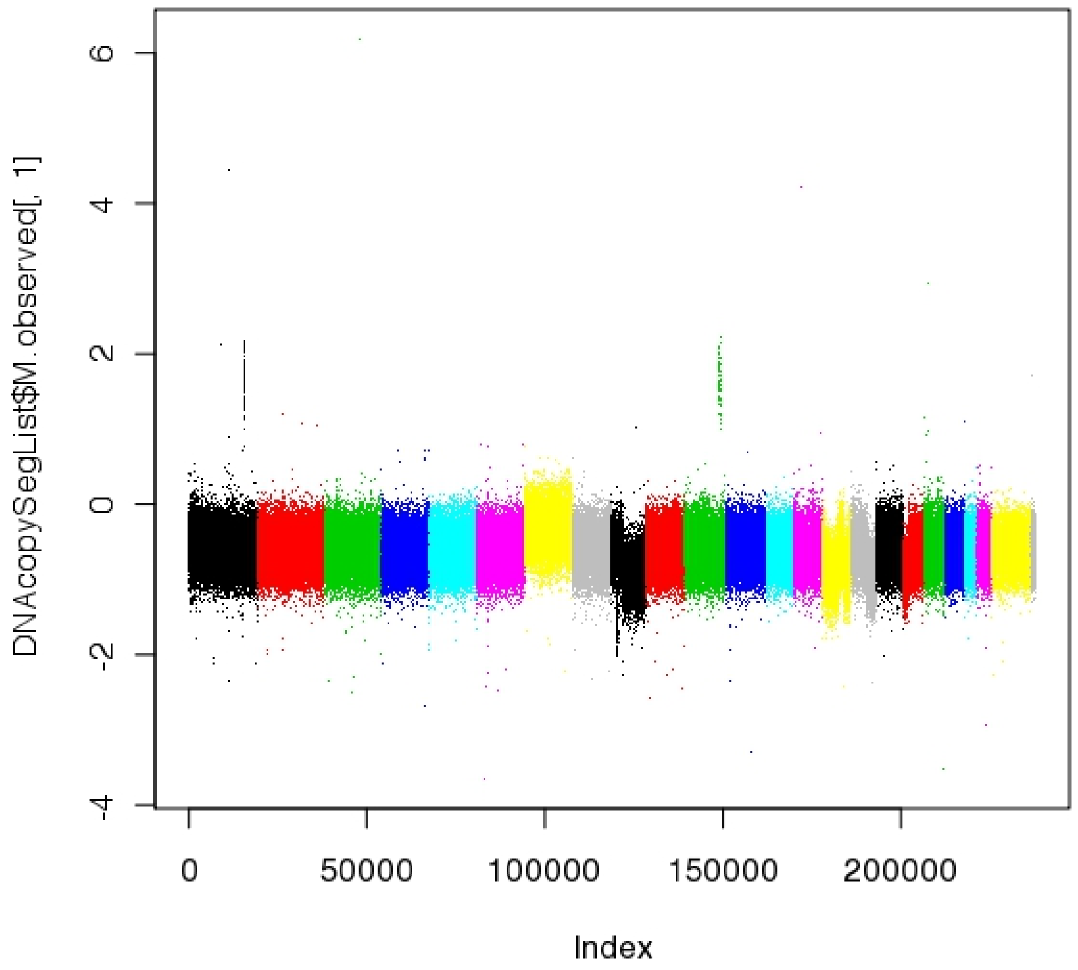
Remember that our goal is to find regions of gain and loss. To do so, we want to make sure that the “baseline” for all the samples is near a log2 ratio of zero. I include “baseline” in quotes because the baseline represents our estimate of the “normal” copy number state. An assumption that I and others will make is that the best estimate of the “normal” state is that which minimizes the number of probes that are not near the log2 ratio of 0.

Perhaps a visual representation is useful here. For this part of the exercise, we will be using DNAcopy segmented values that I prepared earlier. So, load the data:

```
> data(DNAcopySegList)
> dim(DNAcopySegList)
[1] 237834     45
> class(DNAcopySegList)
[1] "SegList"
attr(,"package")
[1] "snapCGH"
```

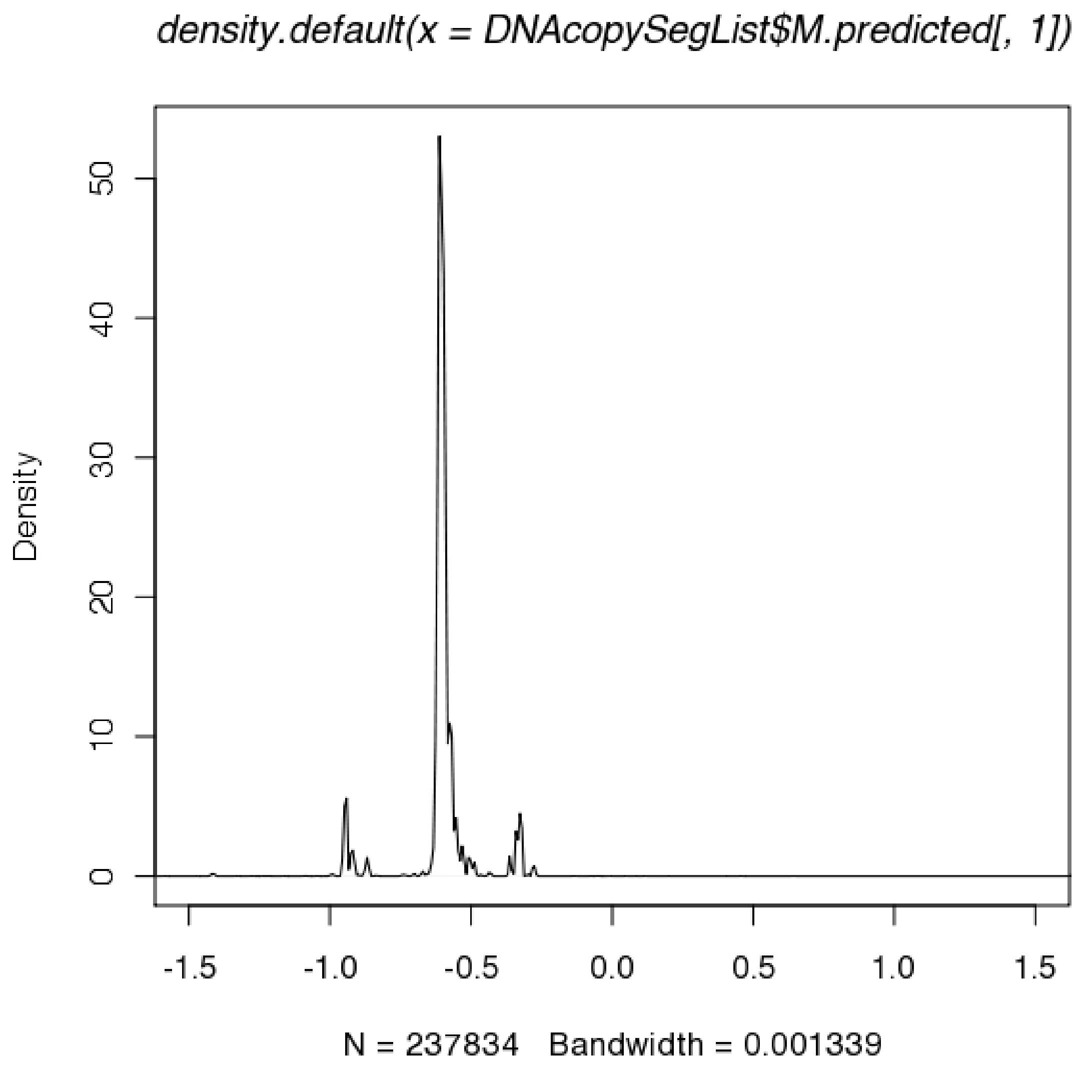
And plot one example where the center is probably not correct:

```
> plot(DNAcopySegList$M.observed[, 1], pch = ".",
+     col = DNAcopySegList$genes$Chr)
```



Notice that the "center" of the distribution is at log2 ratio of -0.6. How can we reliably and robustly find the "center" of this distribution? Let's look at a density plot of the *segmented* data.

```
> plot(density(DNAcopySegList$M.predicted[,
+     1]), xlim = c(-1.5, 1.5))
```



The mode of this distribution is exactly what we are interested in using as the “center”. Therefore, I include a little function to find the mode of a vector of numbers.

```
> findMode = function(z) {
+     tmpdens = density(z)
+     return(tmpdens$x[which.max(tmpdens$y)])
+ }
```

We can apply it to all samples simultaneously using the apply functionality of R.

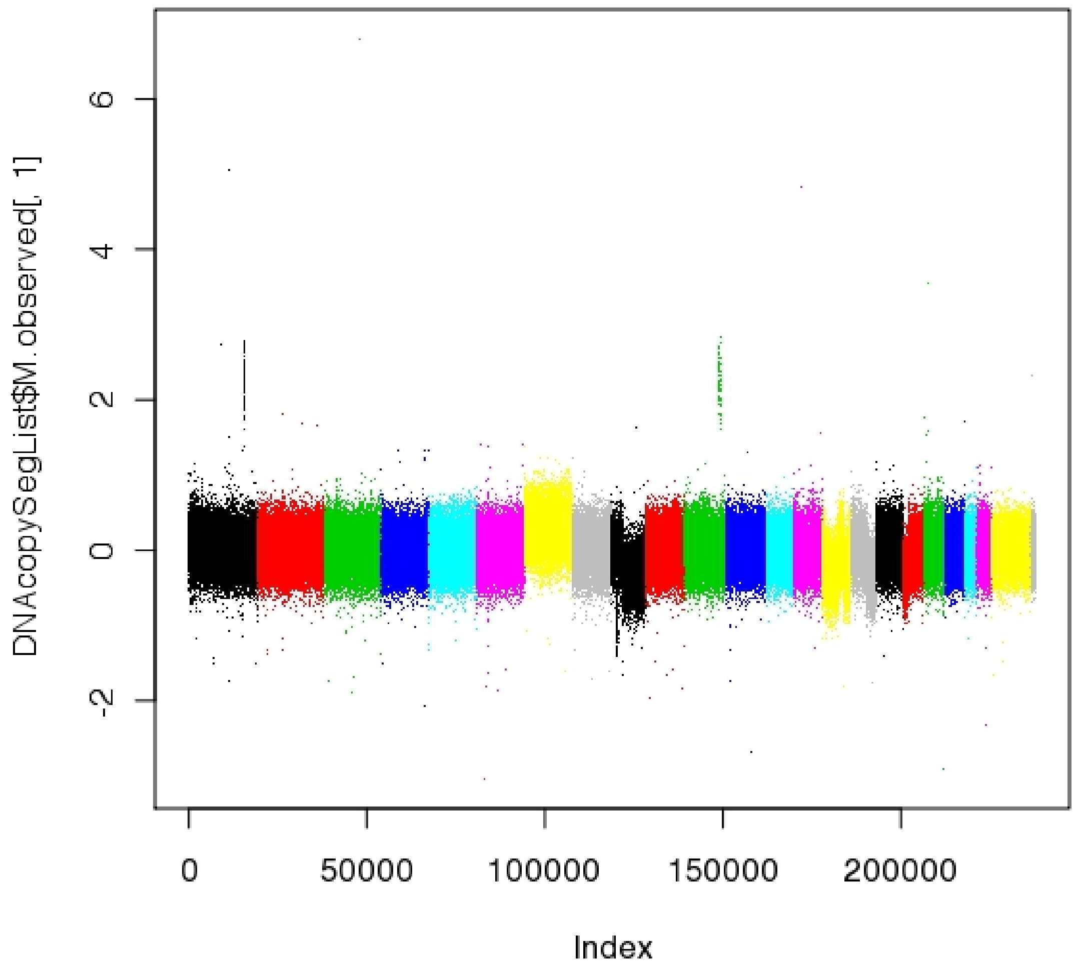
```
> ctrs = apply(DNAcopySegList$M.predicted,
+     2, findMode)
> summary(ctrs)
    Min.  1st Qu.   Median     Mean  3rd Qu.
-0.71250 -0.28210 -0.12500 -0.16040 -0.02486
    Max.
 0.16420
```

To “fix” the data (that is, subtract the offsets from the data to appropriately “center” them), we can use the sweep function to add the appropriate offset to the data to get the “baseline” near the log2 of 0, our goal.

```
> DNAcopySegList$M.predicted = sweep(DNAcopySegList$M.predicted,
+     2, ctrs)
> DNAcopySegList$M.observed = sweep(DNAcopySegList$M.observed,
+     2, ctrs)
```

Now, if we plot the same sample as before, it should show a baseline close to log2 of 0.

```
> plot(DNAcopySegList$M.observed[, 1], pch = ".",
+     col = DNAcopySegList$genes$Chr)
```



Now, on to biology, finally!

Global Copy Number Behavior

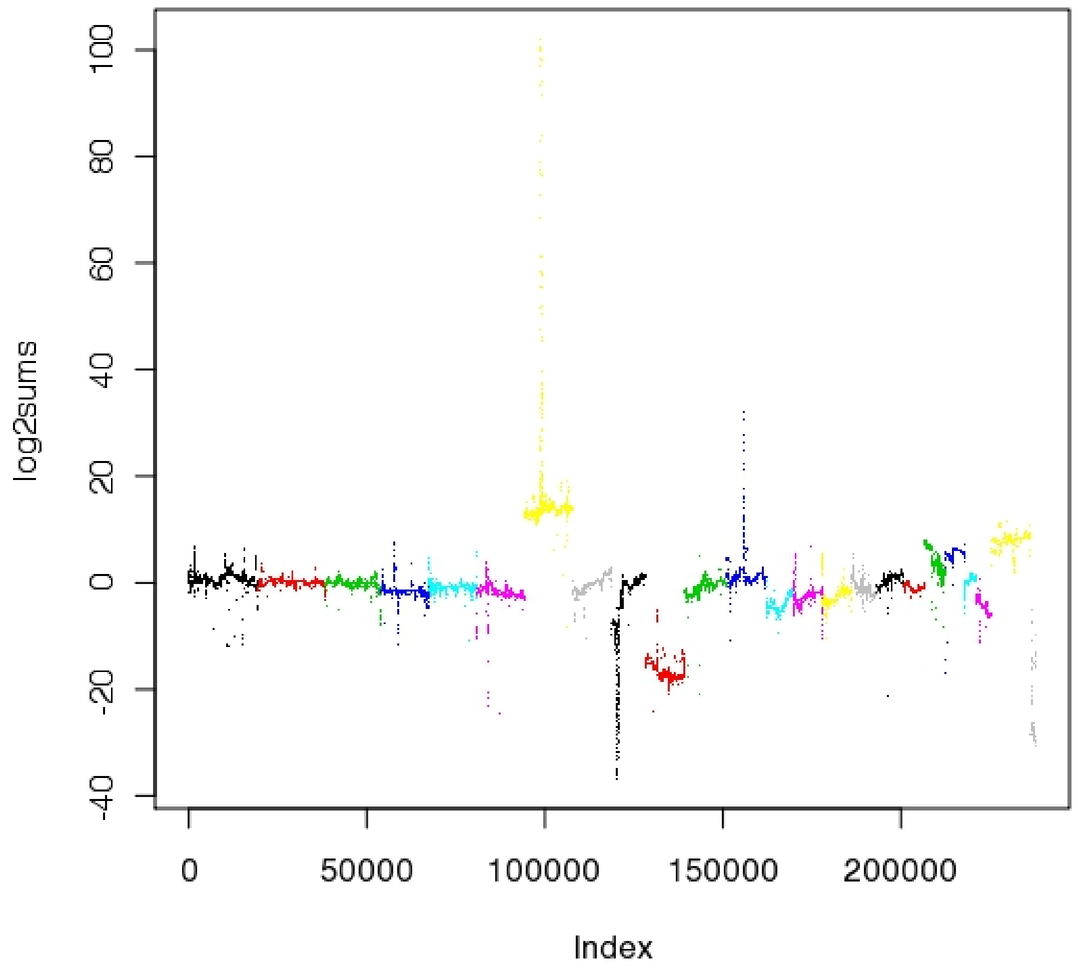
Now that we have performed segmentation, thereby smoothing the data while maintaining the natural breakpoints in the data, we want to look for the regions with the highest proportion of copy number changes in the samples. Particularly if we successfully remove common copy number variants that are present in normal individuals, such regions may be thought to be the most interesting biologically.

There are a number of methods described in the literature to find such "interesting" regions. However, we will keep it simple here and try just a couple. Using our segmented and centered data, let's simply add up the log2 ratios at each locus for every sample.

```
> log2sums = rowSums(DNAcopySegList$M.predicted)
> length(log2sums)
[1] 237834
```

And plot the result.

```
> plot(log2sums, pch = ".", col = DNAcopySegList$genes$Chr)
```



This figure is very important to understand. The distance away from the y=0 line, here simply the sum of the segmented log2 ratios, is somewhat a measure of biological importance. Recall that regions that are gained may contain oncogenes while regions with loss are potentially harboring tumor suppressors. A few regions of interest immediately pop out. There are small, high excursions on chromosomes 7 and 12. There is a significant low excursion on chromosome 9 (and also chromosome Y, which we will choose to ignore, but why would chromosome Y appear to be deleted?). Also, though lower excursion, the baseline for chromosome 7 shows a gain while the baseline for chromosome 10 shows a loss. Does this make sense biologically given what is known about glioblastoma?

This is all well-and-good, but what genes are in the regions of high gain and loss? Well, let's just ask the data. Let's find the probes that correspond to the regions of highest gain and lowest loss. To do so, I will use a simple threshold and pull out the probes that have log2sums higher than the threshold. In these data, the thresholds are quite easy to choose; this is not, in general, the case.

```
> highcopygenes = DNAcopySegList$genes[log2sums >
+     20, c("Chr", "GeneName")]
> unique(highcopygenes[-grep("chr", highcopygenes$GeneName),
+     ])
       Chr GeneName
```

```
210431    7 FLJ45974
65369     7    VSTM2
174544    7 CR613464
140801    7 BC045679
241146    7   SEC61G
165548    7     EGFR
44942     7   K03193
91766     7   LANCL2
93272     7 AK128355
121710    7     ECOP
125336    7 BC015339
192911    7 FLJ44060
209927    7 BC094796
130609    7   ZNF713
135653    7 CR590495
155510    7   MRPS17
156592    7     GBAS
29164     7     PSPH
210171    7    CCT6A
225314    7    SUMF2
119849    7    PHKG1
1910      7   Y10275
190999    7   CHCHD2
128900    7 AL713776
37842     7 BC035176
193160   12      OS9
43012    12   CENTG1
205540   12  TSPAN31
72692    12     CDK4
3017     12   MARCH9
67423    12 AK093897
85074    12  CYP27B1
195380   12   METTL1
155592   12  FAM119B
158880   12     TSFM
135524   12     AVIL
```

For those not in the cancer field, I point out two important cancer genes. The high copy number region on chromosome 7 is harboring an EGFR amplification. Just check the literature for EGFR amplification in glioblastoma! On chromosome 12, CDK4 is in the amplified region and is an important cell cycle gene.

What about the region of loss?

```
> lowcopygenes = DNAcopySegList$genes[log2sums <
+     (-30), c("Chr", "GeneName")]
> unique(lowcopygenes[-grep("chr", lowcopygenes$GeneName),
+     ])
       Chr GeneName
81796    9     MTAP
107171   9 AF109294
166419   9   CDKN2A
41673    9   CDKN2B
10694   24   RBMY1F
```

Again, for non-cancer folks, CDKN2A is a well-known tumor suppressor!

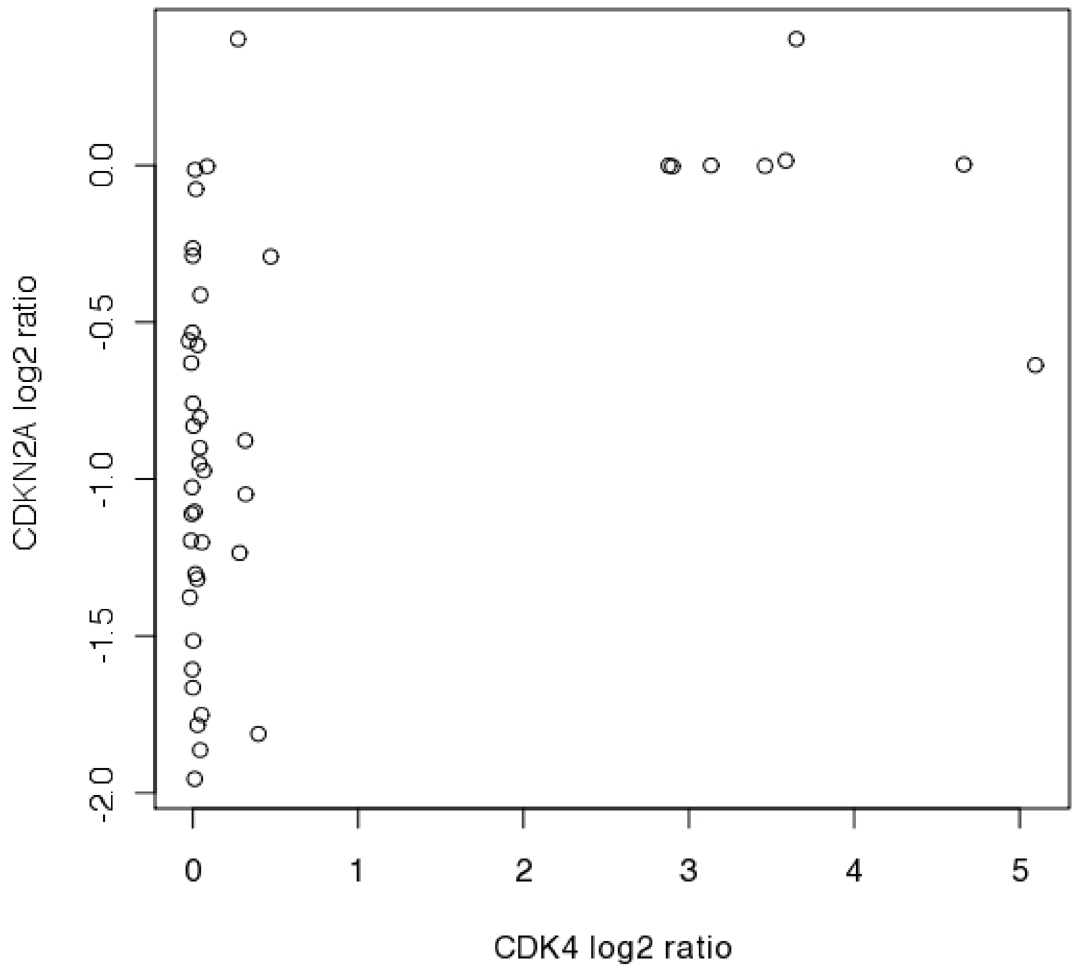
3.3.3 Follow up on interesting finding

So, we have three important cancer genes popping up by basic visual inspection! But, one might begin to wonder a bit about the relationship between these genes. In fact, since there were only three genes that were viable candidates and EGFR was a well-known finding in glioblastoma, I decided to dig a bit further with CDK4 and CDKN2A.

I'll leave it as an exercise to the reader to learn extensively about these two genes and their interactions, but suffice it to say that the full gene name of CDKN2A is "cyclin-dependent kinase inhibitor 2A (melanoma, p16, inhibits CDK4)". CDK4 drives the cell cycle, roughly, and CDKN2A serves to negatively regulate that effect. CDK4 is the prototypic oncogene and CDKN2A is the prototypic tumor suppressor, suppressing tumors by negatively regulating CDK4.

Back to the data, we want to look at the relationship between copy number estimates of CDKN2A and CDK4. I will choose representative probes for each gene and look at a plot of them together.

```
> cdk4probeidx = which(DNAcopySegList$genes$GeneName ==
+     "CDK4")[1]
> cdkn2aprobeidx = which(DNAcopySegList$genes$GeneName ==
+     "CDKN2A")[1]
> plot(DNAcopySegList$M.predicted[cdk4probeidx,
+     ], DNAcopySegList$M.predicted[cdkn2aprobeidx,
+     ], xlab = "CDK4 log2 ratio", ylab = "CDKN2A log2 ratio")
```



How would you interpret the plot? Does this make sense given what you know about CDKN2A and CDK4 biology?

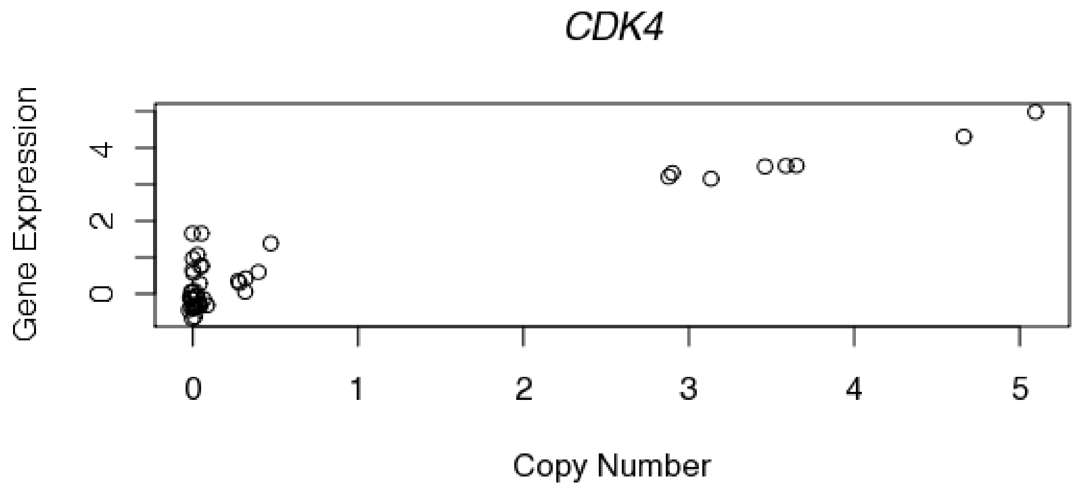
Perhaps we want to look at the effect of copy number on gene expression for these genes. To do so, we need the copy number data to be in the same order as the gene expression data.

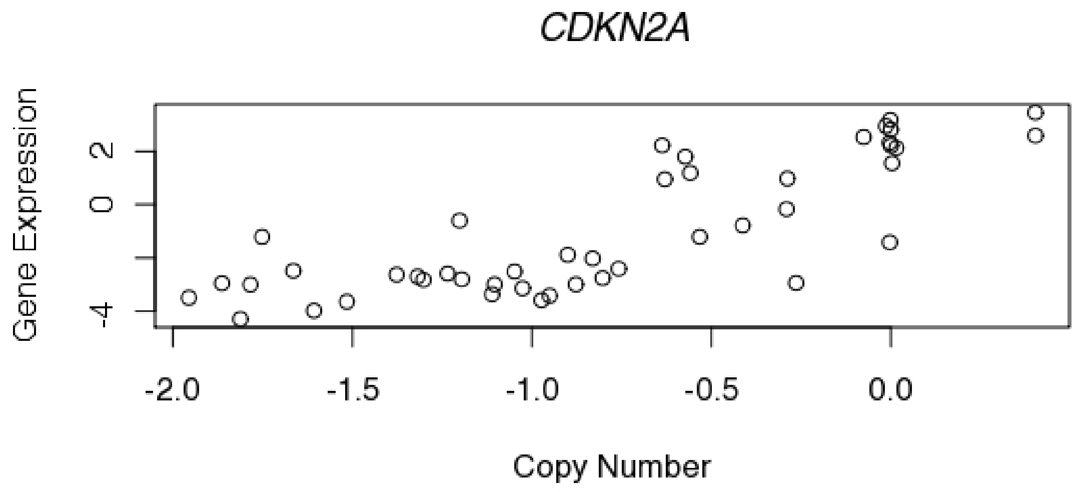
```
> cnCDK4 = DNAcopySegList$M.predicted[cdk4probeidx,
+     order(cghTCGAMA$targets$BCRPATIENTBARCODE)]
> cnCDKN2A = DNAcopySegList$M.predicted[cdkn2aprobeidx,
+     order(cghTCGAMA$targets$BCRPATIENTBARCODE)]
> exCDK4 = exprs(expTCGA)["CDK4", order(sampleNames(expTCGA))]
> exCDKN2A = exprs(expTCGA)["CDKN2A", order(sampleNames(expTCGA))]
```

And now, we can make a plot of the two genes to see how things look:

```
> par(mfrow = c(2, 1))
> plot(cnCDK4, exCDK4, main = "CDK4", xlab = "Copy Number",
+     ylab = "Gene Expression")
```

```
> plot(cnCDKN2A, exCDKN2A, main = "CDKN2A",
+     xlab = "Copy Number", ylab = "Gene Expression")
```





And how about methylation, particularly of CDKN2A, since that could be a secondary mechanism for silencing the gene, besides gene deletion? We'll need to get at least one methylation data vector or CDKN2A.

```
> methGeneNames = unlist(mget(featureNames(methTCGA),
+     IlluminaHumanMethylation27kSYMBOL,
+     ifnotfound = NA))
> cdkn2amethidx = which(methGeneNames ==
+     "CDKN2A")
> cor(t(betas(methTCGA)[cdkn2amethidx, ]))
           cg00718440 cg03079681 cg13479669
cg00718440  1.0000000 -0.1392268 -0.1244518
cg03079681 -0.1392268  1.0000000  0.7301408
cg13479669 -0.1244518  0.7301408  1.0000000
cg26673943 -0.1494681  0.8510897  0.7594127
```

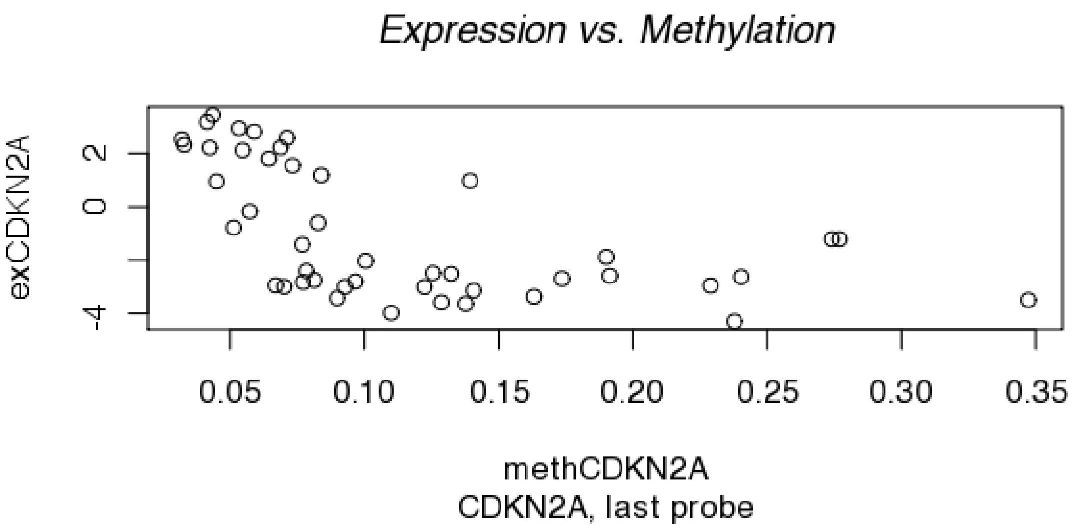
```
           cg26673943
cg00718440 -0.1494681
cg03079681  0.8510897
cg13479669  0.7594127
cg26673943  1.0000000
```

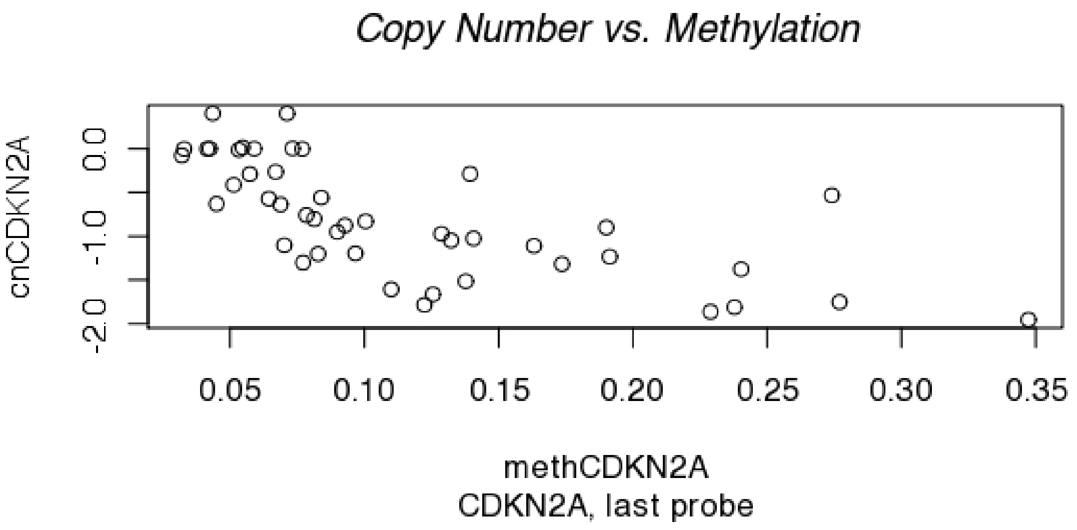
Note that three out of four probes are correlated positively with each other. So, we can exclude the first probe because it does not agree with the other three. This might be an incorrect assumption, but we can always check later. Using the last probe, arbitrarily, as representative, we can now interrogate the effect of methylation on gene expression of CDKN2A.

```
> methCDKN2A = betas(methTCGA)["cg26673943",
+     order(sampleNames(methTCGA))]
```

Finally, let's plot against gene expression and copy number:

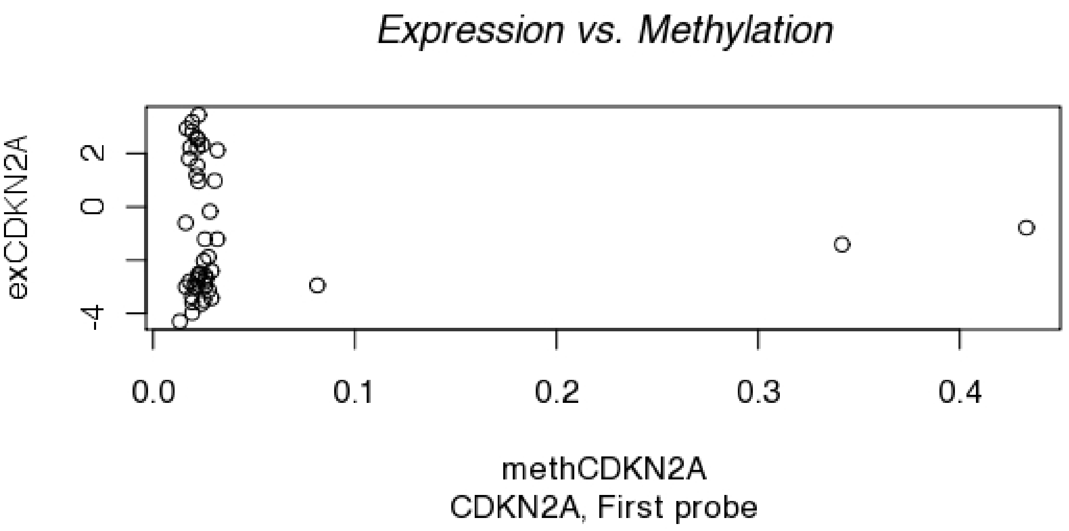
```
> par(mfrow = c(2, 1))
> plot(methCDKN2A, exCDKN2A, main = "Expression vs. Methylation",
+     sub = "CDKN2A, last probe")
> plot(methCDKN2A, cnCDKN2A, main = "Copy Number vs. Methylation",
+     sub = "CDKN2A, last probe")
```

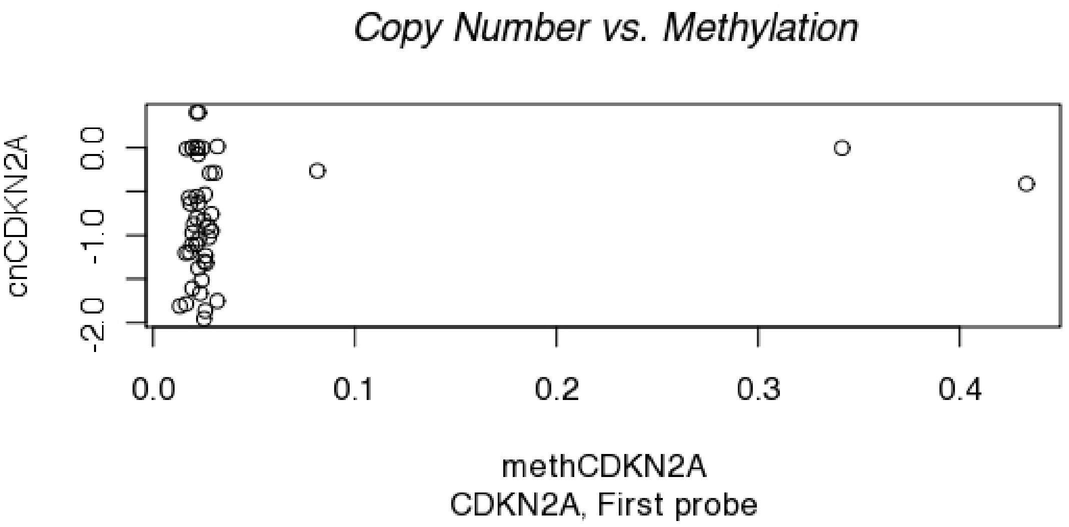




Taking the expression plot, first, we see a very nice negative correlation between expression and methylation, despite the fact that the maximal observed methylation is 0.3 or so. Looking at the bottom plot, however, it appears that the same correlation holds with copy number. We might hope to see that the normal copy number samples would be more likely to have higher methylation values. Instead, the opposite is true. Remember that we chose to ignore the first methylation probe because it was poorly correlated with the other three methylation probes. Let's revisit that probe.

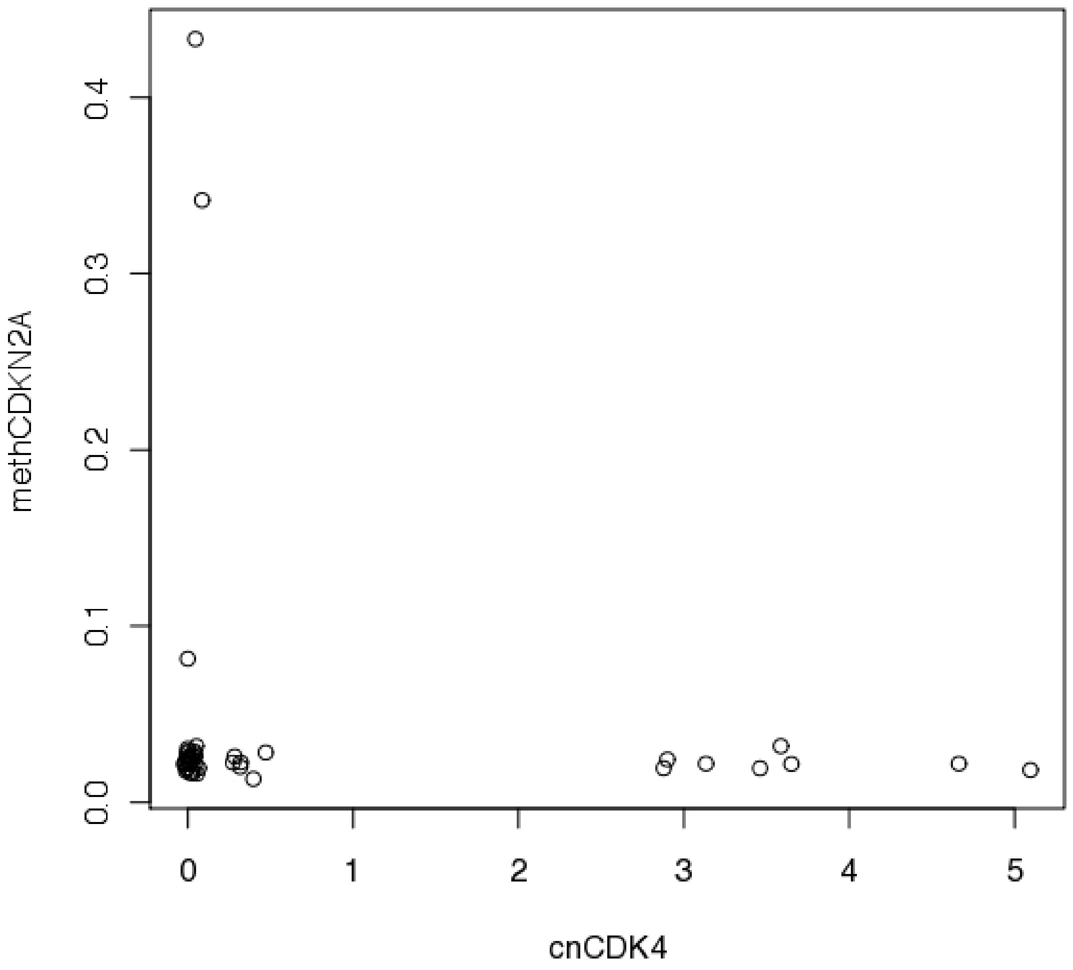
```
> methCDKN2A = betas(methTCGA)["cg00718440",
+     order(sampleNames(methTCGA))]
> par(mfrow = c(2, 1))
> plot(methCDKN2A, exCDKN2A, main = "Expression vs. Methylation",
+     sub = "CDKN2A, First probe")
> plot(methCDKN2A, cnCDKN2A, main = "Copy Number vs. Methylation",
+     sub = "CDKN2A, First probe")
```





While the number of samples with high methylation is small (probably just 2), these samples show lowish CDKN2A gene expression levels. With regard to copy number, the samples with high methylation are, indeed, those without deletion, suggesting that methylation could be responsible for CDKN2A silencing in these samples. And what about the CDK4 amplification status of these samples?

```
> plot(cnCDK4, methCDKN2A)
```



These two samples have stone-cold normal CDK4 copy number.

3.4 Conclusions

We have performed a subset of a data integration exercise on TCGA glioblastoma data and have an incredibly interesting finding within the space of an afternoon. There is a small but measurable global effect of methylation on gene expression. Also, we find in the copy number data the most common amplification, EGFR, in these data as well as gain of chromosome 7 and loss of chromosome 10, also common findings in glioblastoma.

At a more mechanistic level, it seems that the relationship between CDKN2A, acting as tumor suppressor and it's target, CDK4 is quite clear in these data. We have a working hypothesis that CDK4 overexpression due to amplification and CDKN2A underexpression due to either DNA loss (deletion) or methylation at one CpG site are approximately mutually exclusive and likely represent the same pathway being disregulated in these data.

3.5 sessionInfo

```
> sessionInfo()
R version 2.12.0 Under development (unstable) (2010-04-30 r51866)
i386-apple-darwin9.8.0

locale:
[1] en_US/en_US/C/C/en_US/en_US

attached base packages:
[1] stats     graphics  grDevices datasets
[5] utils     methods   base

other attached packages:
 [1] snapCGH_1.19.0
 [2] DNAcopy_1.23.3
 [3] IlluminaHumanMethylation27k.db_1.2.0
 [4] org.Hs.eg.db_2.4.1
 [5] RSQLite_0.9-1
 [6] DBI_0.2-5
 [7] AnnotationDbi_1.11.0
 [8] TCGAGBM_1.0
 [9] limma_3.5.10
[10] methylumi_1.3.3
[11] Biobase_2.9.0
[12] ascii_0.6-4
[13] proto_0.3-8

loaded via a namespace (and not attached):
 [1] aCGH_1.27.0           affy_1.27.0
 [3] affyio_1.17.0         annotate_1.27.0
 [5] cluster_1.12.3        genefilter_1.31.0
 [7] GLAD_2.11.0           grid_2.12.0
 [9] lattice_0.18-8        MASS_7.3-6
[11] multtest_2.5.0        preprocessCore_1.11.0
[13] RColorBrewer_1.0-2    splines_2.12.0
[15] strucchange_1.4-0     survival_2.35-8
[17] tilingArray_1.27.1    tools_2.12.0
[19] vsn_3.17.1            xtable_1.5-6
```

PDF version of course materials